

# QUALITATIVE SPATIAL QUERY PROCESSING

Towards Cognitive Geographic Information Systems

**Rami Al-Salman**

Dissertation  
zur Erlangung des Grades eines Doktors der  
Ingenieurwissenschaften

— Dr.-Ing. —

VORGELEGT IM FACHBEREICH 3 (MATHEMATIK & INFORMATIK)

UNIVERSITÄT BREMEN

August 2014

Datum des Promotionskolloquiums:

Gutachter: Prof. Christian Freksa, Ph.D. (Universität Bremen)  
Prof. Christian S. Jensen, Ph.D. (Aalborg University)

## Abstract

For a long time, Geographic Information Systems (GISs) have been used by GIS-experts to perform numerous tasks including way-finding, mapping, and querying geo-spatial databases. The advancement of Web 2.0 technologies and the development of mobile-based device applications present an excellent opportunity to allow the public —non-expert users— to access information of GISs.

However, the interfaces of GISs were mainly designed and developed based on quantitative values of spatial databases to serve GIS-experts, whereas non-expert users usually prefer a qualitative approach to interacting with GISs. For example, humans typically resort to expressions such as “the building is *near* a riverbank” or “there is a restaurant *inside* a park” which qualitatively locate the spatial entity with respect to another. In other words, the users’ interaction with current GISs is still not intuitive and not efficient. This dissertation thusly aims at enabling users to intuitively and efficiently search spatial databases of GISs by means of qualitative relations or terms such as *left*, *north of*, or *inside*. We use these qualitative relations to formalise so-called Qualitative Spatial Queries (QSQs).

Aside from existing topological models, we integrate distance and directional qualitative models into Spatial Data-Base Management Systems (SDBMSs) to allow the qualitative and intuitive formalism of queries in GISs. Furthermore, we abstract binary Qualitative Spatial Relations (QSRs) covering the aforementioned aspects of space from the database objects. We store the abstracted QSRs in a Qualitative Spatial Layer (QSL) that we extend into current SDBMSs to avoid the additional cost of the abstraction process when dealing with every single query. Nevertheless, abstracting the QSRs of QSL results in a high space complexity in terms of qualitative representations.

Hence, we apply two data reduction strategies so the QSL memory overhead is drastically reduced: (1) reduction by clustering and (2) reduction by a converse operation. The first strategy applies clustering approach to reduce the total space complexity compared to the original size of the QSL. The second strategy applies the converse operation of a qualitative model, to exploit symmetry in the QSL and thus to reduce the size of the QSL.

We consider the spatial query-answering problem as a sub-graph isomorphism matching problem which is NP-complete. To cope with the complexity of a sub-graph isomorphism matching problem, we propose five novel database indexing approaches. The first approach is called Hybrid Interpretation Tree and B<sup>+</sup>-Tree (HITBT) and aims to reduce the time complexity of processing QSQs. As applying HITBT brings a high dimensionality problem in terms of database indexing, we propose Qualitative Hash Table Indexing (QHTI). QHTI concatenates the labels of pairs of objects with their relations and then stores them in a hash table. As space demands of QHTI are high, we propose Qualitative Hash Table Compression (QHTC) as an extension of QHTI. QHTC processes QSQs even quicker than QHTI and at the same time saves space by aggregating the multiple recurrences of data sets induced by QHTI. We develop QHTC of Qualitative Models and QHTC of Object Pairs as variants of QHTC to increase the possibility of find recurring sets induced by QHTI.

We develop **QualEnabler** system that combines the aforementioned components of our work such as QSL, clustering, indexing, etc. In addition, we show the applicability of our system by implementing two prototypical query systems.

Based on **QualEnabler**, we conduct two types of evaluations on real-world and synthetic datasets to evaluate space and time scalability of our approaches. Regarding space scalability, the results of the experiments and their analyses suggest that the data reduction strategies have an ability to reduce the amounts of qualitative representations of QSL significantly. Regarding time scalability, the results of our experiments suggest that QHTI and QHTC are the most scalable approaches in comparison to others to process QSQs.



## Zusammenfassung

Geographische Informationssysteme (GIS) werden seit geraumer Zeit von Experten zur Bewältigung verschiedener Aufgaben eingesetzt, darunter insbesondere zur Beantwortung von Anfragen auf Geodatenbanken. Die rasante Entwicklung der Web-2.0-Technologien in den letzten Jahren und der technische Fortschritt mobiler Geräte ebneten den Weg für neue GIS-basierte Anwendungen, die nun für jedermann verfügbar sind.

Die Schnittstellen von GIS wurden in erster Linie für GIS-Experten entworfen und entwickelt und basieren daher hauptsächlich auf den quantitativen Werten aus den Geodatenbanken. Gewöhnliche Benutzer bevorzugen allerdings eher den qualitativen Ansatz, anstatt mit einem GIS direkt zu interagieren. Beispielsweise verwenden Menschen Ausdrücke, wie “das Gebäude ist in der Nähe des Flussufers” oder “es gibt ein Hotel innerhalb des Parks”, die verschiedene Geo-Objekte in Relation zueinander setzen. Dementsprechend ist die Benutzer-Interaktion bei traditionellen GIS Systemen noch nicht intuitiv und leistungsfähig. Im Rahmen dieser Dissertation soll die beschriebene Lücke geschlossen werden, indem einem Nutzer des GIS die Möglichkeit gewährt wird, Anfragen auf Geodatenbanken intuitiv mit Hilfe von qualitativen Beziehungsbeschreibungen - ausgedrückt durch Relationsdeskriptoren wie “links”, “nördlich”, oder “innen” - zu spezifizieren. Diese qualitativen Deskriptoren können verwendet werden, um sogenannte Qualitative räumliche Anfragen (Qualitative Spatial Queries, QSQs) zu formalisieren.

Neben den bereits bestehenden topologischen Modellen, integrieren wir auf Distanz und Richtung basierende qualitative Modelle in das Spatial-Database-Management-System (SDBMS). Desweiteren extrahieren wir binären qualitative räumliche Deskriptoren (Qualitative Spatial Relation, QSRs), welche die obengenannten räumlichen Aspekte der Datenbankobjekte umfassen. Die abstrahierte QSRs wird in einer Qualitativen räumlichen Ebene (Qualitative Spatial Layer, QSL) gespeichert und dient als Erweiterung der aktuellen

SDBMSe, mit der der zusätzliche Aufwand der Abstraktion jeder einzelnen Anfrage umgangen wird.

Das Ablegen der qualitativen Informationen in QSL ist sehr speicherintensiv und deshalb werden zwei Datenreduktionsstrategien vorgestellt, die den Speicheraufwand erheblich reduzieren: (1) Reduktion durch Clustering und (2) Reduktion durch Anwendung der Gegenoperation. Die erste Strategie wendet ein Clusteringverfahren an, um die Größe der QSL zu reduzieren. Die zweite Strategie erreicht dies durch die Anwendung der gegenteiligen Operation und nutzt damit die Symmetrieeigenschaft von QSL.

Wir betrachten eine räumliche Anfrage auf ein GIS als die Lösung des Subgraph-Isomorphismus-Übereinstimmungsproblems, welches NP-vollständig ist. Um dennoch auf akzeptable Berechnungszeiten zu kommen, stellen wir fünf neue Datenbankindexierungsmethoden vor. Der erste Ansatz wird als Hybrid-Interpretation-Tree und B<sup>+</sup>-Tree (HITBT) bezeichnet und verfolgt in erster Linie die Reduktion der Zeitkomplexität der QSQs-Verarbeitung. Da die Anwendung von HITBT das Problem der hohen Dimensionalität bezogen auf die Indexierung mit sich bringt, stellen wir als zweites das Qualitative-Hash-Table-Indexing (QHTI) Verfahren vor. QHTI konkateniert die Bezeichner von Objektpaaren mit ihren qualitativen Deskriptoren und legt diese anschließend in einer Hashtabelle ab. Da QHTI besonders speicherintensiv ist, wird Qualitative-Hash-Table-Compression (QHTC) entwickelt, ein Verfahren basierend auf einer komprimierten Hashtabelle, welche eine Erweiterung von QHTI darstellt. QHTC benötigt nicht nur weniger Speicher, sondern auch weniger Rechenzeit für die Berechnung von QSQs durch Aggregation von häufig auftretenden Datenwerten erzeugt durch QHTI. Wir haben sowohl die QHTC von qualitativen Modellen als auch die QHTC von Objektpaaren als Varianten von QHTC entwickelt, um die Wahrscheinlichkeit zu erhöhen, dass wiederkehrende Datenwerte gefunden werden.

Wir stellen das QualEnabler-System vor, welches die zuvor genannten Komponenten dieser Arbeit, wie QSL, Clustering, Indexierung, etc., umfasst. Zusätzlich zeigen wir die Praxistauglichkeit unseres Systems durch Implementierung zweier prototypischer Anfragesysteme.

Es wurden zwei Arten von Evaluationen auf einer realen und einer synthetischen Datenmenge durchgeführt, um die Zeit- und Speicherplatzskalierbarkeit des Ansatzes zu ermitteln. Bezogen auf den Speicherplatzverbrauch, haben die Experimente und deren Analysen gezeigt, dass die Datenreduktionsverfahren die Fähigkeit haben die Datenmenge der qualitativen Representation von QSL signifikant zu verkleinern. Hinsichtlich der Zeitkomplexität, haben QHTI und QHTC sich als die effizientesten Verfahren zur Berechnung von QSQs erwiesen.

---

To my mother and father for their love and support. All is due to their  
hard work and sacrifice...

## Acknowledgements

All praise is due to ALLAH (GOD) for pursuing this Ph.D. degree among other endless blessings on me.

I am grateful to acknowledge the European Commission Initial Training Network geocrowd (FP7 - People Marie Curie Actions) and the German Research Foundation (DFG) via the Transregional Collaborative Research Center on Spatial Cognition SFB/TR8 for the financial funds which allow me to pursue my Ph.D. studies.

I would like to thank Prof. Christian Freksa for supervising my dissertation and for giving me the freedom to select my Ph.D. topic and to develop my research ideas. Thank you very much Christian for your very constructive comments and guidelines without them I could not have done my dissertation. Thanks a lot for reserving time for me to do meetings and discussions, although your time was always tight. Every single word you said to me, has had a big impact on my research and communication skills. You were always ready to support me and to push me forward to continue my Ph.D. studies. All these favors and memories will reside in my heart and mind forever. Many thanks also to my co-supervisor Prof. Christian S. Jensen for his valuable comments and suggestions which make my dissertation work more professional. From you I learned how to develop my ideas in a very professional and critical way.

Dr. Frank Dylla your comments were very harshly on me during my Ph.D. studies, but at the same time they were (and they are still) very helpful and significant to develop my thinking way and ideas that led to write this dissertation. Every single meeting we had, it adds in a way something to my research skills and to my dissertation. I learned from you how to be picky in analyzing and discussions, to write academic papers, and to do the

research in a very professional way. How many thankful words I would give to you, they are still not enough for your great efforts in making me a real researcher. It was really a great pleasure to work with you.

Many thanks go to my graduate seminar reviewers, especially Dr. Thomas Barkowsky and Dr. Carl Schultz who were always giving me very valuable comments regarding my research and dissertation.

I would love to thank all reviewers who have reviewed the chapters of my dissertation namely: Dr. Thomas Barkowsky, Dr. Carl Schultz, Dr. Holger Schultheis, Dr. Frank Dylla, Dr. Mohammad Friwan, and Dr. Jae Hee Lee. Special thanks go to Dr. Sadet Alci for his support and great efforts in revising the German abstract of this dissertation.

I am very thankful to geocrowd people who I spent very nice summer schools, meetings, and workshops with them. I have really enjoyed a lot being with you and sharing with you my ideas and thoughts.

Now it is a time to thank my friends who I have spent a lot of my time with them: Dr. Zoe Falomir Llansola, Chunyuan Cai, Ahmed Loai Ali, Dr. Paolo Fogliaroni, and Dr. Giorgio de Felice.

The turn of my family comes. Thank you a lot my mother and father for your sacrifice and unlimited supports to me. I cannot find enough words to thank you, but be sure that I will be supporting for you until the rest of my life. My sister Areen, my brother Mahmoud, my little brother Mostafa, you gave a lot of your support and happiness that pushed me to finish my dissertation. So, thank you very much.

Marianna Sakharova, I have to thank you very much for your endless supports and your encouraging words which helped me a lot to write my dissertation. Without such a great support I cannot imagine that this dissertation can be done.

In the end, I would like to ask everyone who I did not mention here to forgive me; all of you are important to me. But do not worry, although the text space is limited, my heart is big enough to include you all.





# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xv</b>
<b>List of Algorithms</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Application Scenarios . . . . .	6
1.2.1 Daily Life . . . . .	6
1.2.2 Disaster Management . . . . .	6
1.3 Dissertation Hypotheses and Contributions . . . . .	7
1.4 Organization of the Dissertation . . . . .	8
<b>2 Qualitative Spatial Representation and Reasoning</b>	<b>9</b>
2.1 Qualitative Spatial Representations . . . . .	10
2.1.1 Qualitative Spatial Calculi and their Operations . . . . .	11
2.2 Aspects of Qualitative Spatial Calculi . . . . .	12
2.2.1 Topology . . . . .	12
2.2.2 Direction and Orientation . . . . .	14
2.2.3 Distance . . . . .	17
2.2.4 Other Aspects of Qualitative Spatial Calculi . . . . .	18
2.3 Qualitative Constraint Networks . . . . .	18
2.4 Conceptual Neighbourhood-Based Reasoning . . . . .	19

## CONTENTS

---

2.5	Consistency and Relaxations . . . . .	20
2.6	Summary . . . . .	20
<b>3</b>	<b>Spatial Information Management and Systems</b>	<b>21</b>
3.1	Spatial Data-Base Management Systems . . . . .	21
3.1.1	Spatial Queries . . . . .	22
3.1.1.1	Spatial Data Models . . . . .	22
3.1.1.2	Spatial Query Languages . . . . .	26
3.1.2	Spatial Indexing . . . . .	27
3.1.2.1	B-trees . . . . .	27
3.1.2.2	R-trees . . . . .	31
3.1.2.3	Hashing . . . . .	32
3.1.3	Indexing Applications for Spatial Databases . . . . .	33
3.2	Clustering . . . . .	35
3.2.1	Grid-Based Clustering . . . . .	36
3.2.2	Density-Based Clustering . . . . .	38
3.2.3	Approaches for Clustering Qualitative Data . . . . .	39
3.3	Integrating Qualitative Spatial Reasoning with GISs . . . . .	40
3.3.1	Approaches for Intuitive Interactions . . . . .	40
3.3.2	Approaches for Matching Geo-Spatial Information . . . . .	41
3.4	Summary . . . . .	42
<b>4</b>	<b>Querying, Reducing, and Matching Qualitative Information</b>	<b>45</b>
4.1	Qualitative Spatial Queries . . . . .	45
4.2	Enabling Qualitative Spatial Queries in GISs . . . . .	47
4.3	Extending a Qualitative Spatial Layer into GISs . . . . .	50
4.3.1	Multi-Graph Representations . . . . .	53
4.3.2	Matching a Qualitative Spatial Layer . . . . .	54
4.4	Qualitative Data Reduction . . . . .	56
4.4.1	Qualitative Data Reduction by Clustering . . . . .	56
4.4.1.1	Enclosing Clusters . . . . .	60
4.4.1.2	Qualitative Spatial Relations Between Clusters . . . . .	62
4.4.1.3	Abstracting a Qualitative Spatial Layer From Clusters . . . . .	66
4.4.1.4	Matching the Qualitative Spatial Layer of Clusters . . . . .	67

4.4.2	Qualitative Data Reduction By a Converse Operation . . . . .	69
4.5	Summary . . . . .	72
<b>5</b>	<b>Optimizing Indexing Approaches for Spatial Databases</b>	<b>73</b>
5.1	A Hybrid Interpretation Tree and B <sup>+</sup> -Tree . . . . .	73
5.1.1	Index Construction-HITBT . . . . .	74
5.1.2	Search-HITBT . . . . .	75
5.1.3	Delete-HITBT . . . . .	76
5.1.4	Discussion . . . . .	76
5.2	Qualitative Hash Table Indexing . . . . .	77
5.2.1	Index Construction-QHTI . . . . .	78
5.2.2	Search-QHTI . . . . .	79
5.2.3	Delete-QHTI . . . . .	80
5.2.4	Discussion . . . . .	80
5.3	Qualitative Hash Table Compressing . . . . .	82
5.3.1	Index Construction-QHTC . . . . .	82
5.3.2	Search-QHTC . . . . .	85
5.3.3	Delete-QHTC . . . . .	85
5.3.4	Discussion . . . . .	86
5.4	Qualitative Hash Table Compressing of Qualitative Models . . . . .	88
5.4.1	Index Construction-QHTC <sup>M</sup> . . . . .	88
5.4.2	Search-QHTC <sup>M</sup> . . . . .	90
5.4.3	Delete-QHTC <sup>M</sup> . . . . .	90
5.4.4	Discussion . . . . .	91
5.5	Qualitative Hash Table Compressing of Object Pairs . . . . .	92
5.5.1	Index Construction-QHTC <sup>P</sup> . . . . .	92
5.5.2	Search-QHTC <sup>P</sup> . . . . .	92
5.5.3	Delete-QHTC <sup>P</sup> . . . . .	94
5.6	Summary . . . . .	95
<b>6</b>	<b>Implementation and Applications</b>	<b>97</b>
6.1	PostGIS: A Spatial Layer . . . . .	97
6.1.1	Integrating Qualitative Spatial Models into PostGIS . . . . .	98
6.2	A Qualitative Spatial Layer . . . . .	100

## CONTENTS

---

6.3	DBSCAN Clustering Implementation . . . . .	100
6.4	Indexing Approaches Implementation . . . . .	102
6.4.1	A Hybrid Interpretation Tree and B <sup>+</sup> -Tree . . . . .	102
6.4.2	A Qualitative Hash Table Indexing . . . . .	103
6.4.3	Qualitative Hash Table Compression . . . . .	104
6.4.4	The QHTC of Qualitative Models . . . . .	104
6.4.5	The QHTC of Object Pairs . . . . .	105
6.5	Client-Side Interfaces . . . . .	105
6.5.1	Web-Based Interfaces . . . . .	106
6.5.2	Android-Based Interfaces . . . . .	109
6.6	System Evaluation . . . . .	110
6.7	Summary . . . . .	112
<b>7</b>	<b>Empirical Evaluation</b>	<b>115</b>
7.1	Clustering Experiments . . . . .	115
7.1.1	The Experimental Settings of Clustering . . . . .	116
7.1.2	Filtering Clustering Candidates . . . . .	117
7.1.3	Selecting Clustering Candidate . . . . .	123
7.2	Indexing Approaches Experiments . . . . .	128
7.2.1	The Experimental Settings of Indexing Approaches . . . . .	128
7.2.2	Qualitative Data Reduction . . . . .	131
7.2.3	Varying the Number of Queries . . . . .	132
7.2.4	Varying the Number of Pairs . . . . .	135
7.2.5	Varying the Number of Objects . . . . .	138
7.3	Synthetic Data Evaluation . . . . .	141
7.3.1	Clustering Experiments on a Synthetic Data . . . . .	141
7.3.2	Indexing Approaches Experiments on a Synthetic Data . . . . .	145
7.3.2.1	Qualitative Data Reduction . . . . .	146
7.3.2.2	Varying the Number of Queries . . . . .	147
7.4	Summary . . . . .	149

<b>8</b>	<b>Conclusions</b>	<b>151</b>
8.1	Summary . . . . .	151
8.2	Future Directions . . . . .	154
8.2.1	Qualitative Spatial Clustering Reasoning . . . . .	154
8.2.2	Conceptually Neighboring Qualitative Spatial Queries . . . . .	154
8.2.3	Approximate Qualitative Spatial Query Matching . . . . .	155
8.2.4	Indexing for Qualitative, Spatial, and Keywords Queries . . . . .	155
8.2.5	Supporting Individuals of Qualitative Spatial Queries . . . . .	155
8.2.6	Parallelism of Hash-Based Indexing Approaches . . . . .	156
	<b>References</b>	<b>157</b>
	<b>Appendix A Own Publications</b>	<b>167</b>
A.1	Within the Scope of this Dissertation . . . . .	167
A.2	Out of the Scope of this Dissertation . . . . .	169

## CONTENTS

---

# List of Figures

1.1	Examples of results provided in order to answer the qualitative spatial query “find a restaurant <i>inside</i> a park and <i>near</i> a riverbank”. . . . .	4
2.1	Interior, boundary, and exterior of two regions $A$ and $B$ . . . . .	13
2.2	the eight distinct topological relations from two points sets $(A, B)$ embedded in $\mathcal{D}^2$ with their matrix values. . . . .	14
2.3	(a) the cone-based and (b) the projection-based models. . . . .	15
2.4	A 3x3 matrix is used to represent the binary directional relation(s) between $A$ and $B$ . . . . .	16
2.5	$B_1 \{[NW]\}A$ and $B_2 \{[W, NW, N]\}A$ . . . . .	16
2.6	Two levels of granularity of distance model, adapted from (Hernández et al., 1995). . . . .	18
3.1	The geometry object model, from (Open Geospatial Consortium (OGC) Inc., 2011). . . . .	24
3.2	Example of B-tree and B <sup>+</sup> -tree of order $P = 3$ . The values are inserted in the order $\{1, 2, 3, 4, 5, 6, 7, 8\}$ . . . . .	29
3.3	An example of an R-tree for 2D geometric objects. . . . .	32
3.4	An example: DBSCAN. . . . .	39
4.1	An example: qualitative spatial query formalism. . . . .	47
4.2	An example: verbal descriptions of a QSQ are translated to SQL. . . . .	48
4.3	The distance model with four relations. . . . .	50

## LIST OF FIGURES

---

4.4	A logical view of the qualitative database layer extension. . . . .	51
4.5	Example: a qualitative spatial layer that represents all the binary qualitative spatial relations (per each one of the three qualitative models including topology, direction, and distance) among four geometric objects. . . . .	52
4.6	The $QCN^D$ using three qualitative models T, D, and S. . . . .	53
4.7	Matching $\mathcal{G}_Q$ against $\mathcal{G}_D$ : the first subset is exactly matched by users query, the second subset is partially matched, where the pairs $\{(A, C), (A_1, C_4)\}$ , $\{(B, C), (B_2, C_4)\}$ differ by a directional relation. . . . .	55
4.8	Matching the <i>Itree</i> to the unary and binary constraints of $\mathcal{G}_Q$ . . . . .	56
4.9	A clustering of the objects of Bremen inner city using DBSCAN( $MinPts=2$ , $Eps=300$ ). . . . .	59
4.10	Examples of qualitative data clustering and reduction by DBSCAN: (a) all objects are grouped into a single cluster, (b) all objects are grouped into two clusters, and (c) all objects are grouped into four clusters. . . . .	60
4.11	An example: the MBR, CH, and CCH of a cluster. . . . .	61
4.12	An example: computing <i>disjoint</i> relations between clusters based on three cluster representations the MBR, CH, and CCH, where the yellow color presents the non- <i>disjoint</i> relations. . . . .	63
4.13	(a) shows the transitive relation <i>NE</i> between the reference cluster $C_i$ and the primary cluster $C_j$ , (b) shows the non-transitive relation <i>N</i> between the reference cluster $C_i$ and the primary cluster $C_j$ . . . . .	67
4.14	An example: pruning half of $QCN^D$ space based on symmetry. . . . .	70
5.1	An example: the index construction of the first level of $T^B$ of HITBT. . . . .	74
5.2	Architecture of Qualitative Hash Table Indexing (QHTI) and Compression (QHTC). . . . .	78
5.3	An example: the index construction of the first level of T. . . . .	79
5.4	Example of structuring and matching a query against the second level of T. . . . .	82
5.5	An example: the index construction of the first level of $T^C$ . . . . .	85
6.1	An overview of system architecture. . . . .	98
6.2	Qualitative spatial layer database schema design. . . . .	100



6.3	A snapshot of DBSCAN analyzer; the user needs to specify two parameters: (1) the minimum number of points ( $MinPts$ ) within a cluster and (2) the radius of a cluster ( $Eps$ ). . . . .	101
6.4	DBSCAN database schema design. . . . .	103
6.5	QHTI database schema design. . . . .	104
6.6	QHTC database schema design. . . . .	105
6.7	QHTC <sup>M</sup> database schema design. . . . .	106
6.8	QHTC <sup>P</sup> database schema design. . . . .	107
6.9	The system architecture of the Qualitative Spatial Management System. . . . .	108
6.10	A snapshot of the graphical user interface of QSMS. . . . .	109
6.11	A snapshot of the Android Sketching and Querying Tool. . . . .	110
6.12	System evaluation database schema design. . . . .	111
7.1	A snapshot of the real dataset of Bremen inner city. . . . .	116
7.2	Snapshots of DBSCAN( $MinPts$ , $Eps=v$ ), $v$ : the radius of clusters varied from 50 to 490 meters incremented by 10 meters. . . . .	119
7.3	Snapshots of DBSCAN(2, $Eps$ ): $Eps \in \{50, 100, 290, 300, 310, 490\}$ . . . . .	120
7.4	A snapshot of the CCHs: DBSCAN(2, 300) v.s. DBSCAN(3, 300). . . . .	125
7.5	A snapshot of the MBRs: DBSCAN(2, 300) v.s. DBSCAN(3, 300). . . . .	126
7.6	SQL code: a single pair query with its three spatial relations. . . . .	131
7.7	The space reduction rates of $\mathcal{G}_{\mathcal{D}}$ by QHTC, QHTC <sup>M</sup> , and QHTC <sup>P</sup> . . . . .	131
7.8	Varying the number of single pair queries. . . . .	133
7.9	Varying the number of double pairs queries. . . . .	136
7.10	Comparing our approaches by varying the number of object pairs. . . . .	137
7.11	Comparison: varying the number of objects of the database (cardinality). . . . .	140
7.12	A snapshot of a synthetic dataset. . . . .	142
7.13	Snapshots of DBSCAN( $MinPts$ , $Eps=v$ ), $v$ : the radius of clusters varied from 1 to 30 degrees incremented by 1. . . . .	143
7.14	Varying the number of single pair queries. . . . .	148

## LIST OF FIGURES

---

# List of Tables

2.1	Thirteen spatial relations from (Freeman, 1975). . . . .	12
3.1	Three kinds of spatial operations are provided: (1) basic operators, (2) topological set operators, and (3) spatial analysis operators, from (Open Geospatial Consortium (OGC) Inc., 2011). . . . .	25
3.2	The types of spatial queries, the possible spatial operations, spatial queries, and their spatial indexing methods. . . . .	28
3.3	Comparison between eight categories of clustering methods. . . . .	37
4.1	The distance decisive and indecisive relations. . . . .	65
4.2	Eight binary relations of 9-Intersection-Model and their inverses from (Egenhofer, 1994a). . . . .	70
4.3	Comparison between the three matching approaches. . . . .	72
5.1	Comparison between the five indexing approaches. . . . .	96
6.1	The predicates and their descriptions from <a href="http://postgis.net/docs/manual-2.0/">http://postgis.net/docs/manual-2.0/</a> . . . . .	99
7.1	DBSCAN Parameter settings. . . . .	117
7.2	$r$ and $p$ -value for <i>MinPts</i> with other variables. . . . .	122
7.3	The clustering candidates of DBSCAN experiments. . . . .	122
7.4	Generating the CCHs of clusters using $\alpha$ . . . . .	124

## LIST OF TABLES

---

7.5	The reduction rates for the topological, directional, and distance relations as well as their average reduction. . . . .	126
7.6	The first level of the constructed trees and their index construction time.	130
7.7	The number of abstracted relations of the $C_R$ and $\mathcal{G}_D^C$ . . . . .	130
7.8	Parameter settings. . . . .	131
7.9	The minimum, maximum, average, and standard deviation ( $\sigma$ ) execution time measured by seconds for the spatial queries. . . . .	134
7.10	Four spatial queries and their cardinalities as well as their numbers of the retrieved results. . . . .	139
7.11	$r$ and $p$ -value for <i>MinPts</i> with other variables. . . . .	144
7.12	The clustering candidates of DBSCAN experiments. . . . .	144
7.13	The reduction rates for the topological, directional, and distance relations as well as their average reduction. . . . .	145
7.14	The first level of the constructed trees and their index construction time.	146
7.15	The number of detected unique tuples of QHTC, QHTC <sup>M</sup> , and QHTC <sup>P</sup> as well as their new and reduced graph size. . . . .	147

# List of Algorithms

1	AbstractDataBaseGraph( <i>Objects</i> $O_{\mathcal{D}}$ , <i>ObjGeometries</i> $F_{\mathcal{D}}$ ) . . . . .	52
2	Qualitative_Layer_Matcher( <i>DBgraph</i> $\mathcal{G}_{\mathcal{D}}$ , <i>Query</i> $\mathcal{G}_{\mathcal{Q}}$ ) . . . . .	57
3	Abstract_Relations_Clusters( <i>Clusters</i> $C$ , <i>GeomtryClusters</i> $F$ ) . . . . .	68
4	DBSCAN_Matcher( $\mathcal{G}_{\mathcal{D}}^C$ , $C^R$ , <i>Clusters</i> $C$ , <i>Query</i> $\mathcal{G}_{\mathcal{Q}}$ ) . . . . .	69
5	IndexConstruction_HITBT( <i>DBgraph</i> $\mathcal{G}_{\mathcal{D}}$ , <i>TreeLevel</i> $\ell$ ) . . . . .	75
6	Match_HITBT( $B^+$ -trees $T^B$ , <i>QueryGraph</i> $\mathcal{G}_{\mathcal{Q}}$ ) . . . . .	76
7	Delete_HITBT( $B^+$ tree $T^B$ , <i>Object</i> $o$ ) . . . . .	77
8	IndexConstruction_QHTI( <i>DBgraph</i> $\mathcal{G}_{\mathcal{D}}$ ) . . . . .	80
9	Search_QHTI( <i>QHTI_tree</i> $T$ , <i>Query</i> $\mathcal{G}_{\mathcal{Q}}$ ) . . . . .	81
10	Get( <i>Hash List</i> $HList$ , <i>Hash Key</i> $QHash$ ) . . . . .	81
11	Delete_QHTI( <i>QHTI_tree</i> $T$ , <i>Object</i> $o$ ) . . . . .	83
12	IndexConstruction_QHTC( <i>QHTI_tree</i> $T$ , <i>GeomDB</i> $F_{\mathcal{D}}$ ) . . . . .	84
13	Search_QHTC( <i>QHTC_tree</i> $T^C$ , <i>Query</i> $\mathcal{G}_{\mathcal{Q}}$ , <i>DBPointers</i> $DBh$ , <i>GeomDB</i> $F_{\mathcal{D}}$ ) . . . . .	86
14	Delete_QHTC( <i>QHTC_tree</i> $T^C$ , <i>DBPointers</i> $DBh$ , <i>Object</i> $o$ , <i>GeomDB</i> $F_{\mathcal{D}}$ ) . . . . .	87
15	IndexConstruction_QHTC <sup>M</sup> ( <i>QHTI_tree</i> $T$ ) . . . . .	89
16	Search_QHTC <sup>M</sup> ( <i>QHTC<sup>M</sup>_tree</i> $T^M$ , <i>Query</i> $\mathcal{G}_{\mathcal{Q}}$ ) . . . . .	90
17	Delete_QHTC <sup>M</sup> ( <i>QHTC<sup>M</sup>_tree</i> $T^M$ , <i>Object</i> $o$ ) . . . . .	91
18	IndexConstruction_QHTC <sup>P</sup> ( <i>QHTI_tree</i> $T$ ) . . . . .	93
19	Search_QHTC <sup>P</sup> ( <i>QHTC<sup>P</sup>_tree</i> $T^P$ , <i>Query</i> $\mathcal{G}_{\mathcal{Q}}$ ) . . . . .	94
20	Delete_QHTC <sup>P</sup> ( <i>QHTC<sup>P</sup>_tree</i> $T^P$ , <i>Object</i> $o$ ) . . . . .	95
21	Auto_Queries_Generator( $\mathcal{G}_{\mathcal{D}}$ , $T^C$ , <i>Min</i> , <i>Max</i> , <i>Unq</i> , <i>nq</i> ) . . . . .	112

## LIST OF ALGORITHMS

---

# Chapter 1

## Introduction

In this chapter, we give the motivation of the work presented in this dissertation (Section 1.1). Afterwards, we describe two real life application scenarios (Section 1.2). The dissertation hypotheses and contributions are given in Section 1.3. The last section describes the outline of the dissertation (Section 1.4).

### 1.1 Motivation

Geographic Information Systems (GISs) are computer systems designed to represent, maintain, and analyze spatial data (Worboys and Duckham, 2004). As such, they usually have to cope with huge data sources which must be managed as efficiently as possible and are typically maintained in spatial databases. In particular, spatial databases represent spatial data based on quantitative values which are encoded in either vector or raster format. Efficient methods and algorithms for handling database queries based on these quantitative values were developed. In general, proposed methods focus on two types of queries: (1) path queries (Chen and Xu, 2000; Egenhofer, 1993)(e.g., “find me the shortest path between location A and location B”) and (2) location-based or geo-referenced queries (Jensen et al., 2004a), in which *geographic locations* are usually given. For instance, in a location-based query such as: “from my location: find a **restaurant** within a distance of **400 meters**”, **400** meters is a quantitative value related to the distance of the **restaurant**.

Conversely, humans usually prefer to use qualitative descriptions to communicate geographical information (Mark et al., 1999) such as distance, location, or topology.

## 1. INTRODUCTION

---

For example, humans typically resort to expressions such as “the building is *near* a riverbank” and “there is a restaurant *inside* a park” qualitatively locate a spatial entity with respect to another. Thus, the opportunity to query spatial databases in a qualitative and natural language manner is more intuitive to humans. We therefore need to relate the quantitative data of a given spatial database to qualitative relations used by humans.

Knowledge about spatial configurations can be represented by qualitative spatial relations such as *near*, *far*, etc. for the dimension of distance, and *left*, *ahead*, etc. for the dimension of direction. Other dimensions are, for example, size and topology (Falomir et al., 2013). In the simplest case, these are binary relations that reflect spatial properties for pairs of objects. These relations can be exploited to formalize Qualitative Spatial Queries (QSQs). For instance, in the query “Find a *restaurant near* a *riverbank*”, *near* is a qualitative binary distance relation that holds between some object *restaurant* and some object *riverbank*. The research area of Qualitative Spatial representation and Reasoning (QSR) deals with such sets of binary relations, reasoning operations, and their mathematical properties.

In contrast to other types of queries, here we focus on QSQs which are *non-geo-referenced* queries (e.g., query by natural language or query-by-sketch (Egenhofer, 1997)), in which geographic locations are usually not given. Additionally, QSQs are limited to query categories or classes of objects (e.g., rivers) rather than individuals (e.g., the “Weser” river). Moreover, we only consider objects of atomic categories such as river or building, and no higher order ontological categories such as state or country, which may summarize several atomic objects. Even though QSQs do not usually have geographic locations such as  $53^{\circ}5'N, 8^{\circ}48'E$ , their nature is geographic, spatial, and qualitative. Such queries occur in everyday life and have several applications such as urban planning, disaster management, and services to find locations.

The interaction of users with GISs is not efficient and not intuitive. When users submit their QSQs that contain several binary spatial relations, GISs may fail to answer them efficiently. Consider the following examples in Bremen city: even though there are restaurants in Bremen that are *inside* park(s) and *near* riverbank(s), Figure 1.1 shows



that the three geographic spatial search engines; Google<sup>1</sup>, OpenStreetMap<sup>2</sup>, and Bing<sup>3</sup> fail to answer the query “find a restaurant *inside* a park and *near* a riverbank”. Such queries are not answered satisfactorily due to one or more of the following four reasons:

1) the qualitative spatial representations (e.g., cardinal directions) need to be integrated into Spatial Data-Base Management Systems (SDBMSs) to enable the statement of QSQs in GISs. Hence, in this dissertation we address the following two research questions:

*What are the appropriate qualitative spatial representations that need to be integrated into SDBMSs of GISs?*

*How can the appropriate qualitative spatial representations be integrated into SDBMSs of GISs?*

2) a qualitative abstraction of the quantitative data of spatial databases is crucial to achieve real-time performance. In order to answer QSQs, GISs need to abstract (or compute) qualitative spatial relations in a database at run time, which is computationally infeasible and impractical. To solve this issue we introduce a Qualitative Spatial Layer (QSL) that covers suitable qualitative spatial models (or features) which can first be abstracted, and then extended to the SDBMSs of GISs. Consequently, GISs will avoid the additional cost of the abstraction process every time when answering QSQs. Unfortunately, abstracting the QSL results in a high space complexity in the amounts of qualitative data to be considered when answering QSQs. However, by applying spatial data mining (e.g., clustering) techniques and QSR operations (e.g., composition), the amount of qualitative data in the QSL can be reduced. Based on this discussion, we address the following question:

*How can spatial data mining techniques and QSR operations reduce the amounts of qualitative data in the QSL?*

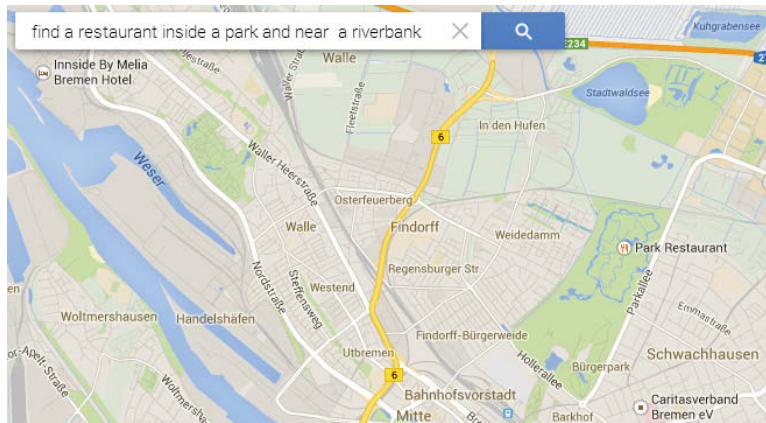
---

<sup>1</sup>Google: <https://maps.google.com>

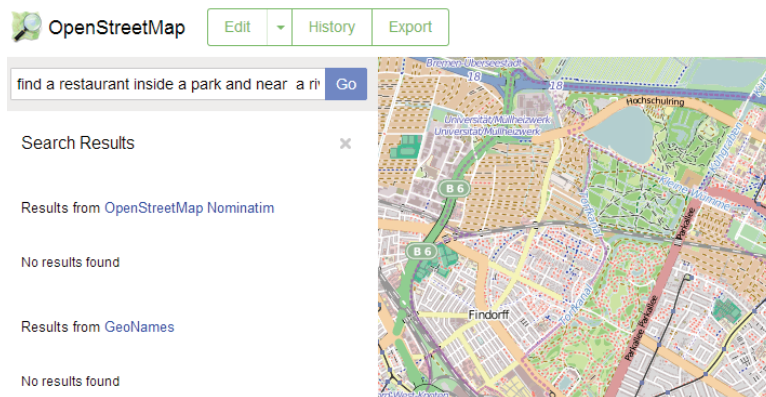
<sup>2</sup>OpenStreetMap: <http://www.openstreetmap.org>

<sup>3</sup>Bing: <http://www.bing.com/maps>

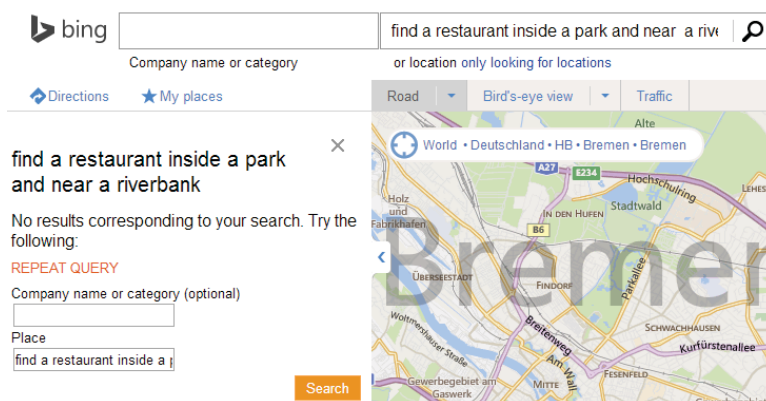
## 1. INTRODUCTION



(a) Google Maps: returns empty result.



(b) OpenStreetMap: returns empty result.



(c) Bing Maps: returns empty result.

**Figure 1.1:** Examples of results provided in order to answer the qualitative spatial query “find a restaurant *inside* a park and *near* a riverbank”.

3) qualitative spatial descriptions and queries can be inaccurate and ambiguous (Wasow et al., 2005) and may lead to misinterpretations. For example, in a spatial query such as “Find a restaurant *near* a riverbank”, the term *near* needs context (Freksa, 1981, pp. 112). It may be interpreted as a distance less than 500 meters, two kilometers, or kilometers depending on the means of transportation (e.g., by foot, bike, or car) and different reference system of each user. Furthermore, the term *restaurant* needs grounding in the quantitative data in order to determine its location and extent. The term *restaurant* needs grounding with respect to the qualitative data to understand/define the meaning of the concept, for example by means of an ontology. Resolving the ambiguity of qualitative descriptions is beyond the scope of this dissertation.

4) answering QSQs in large spatial databases results in a high space and time complexity (Wallgrün et al., 2010). In order to answer queries, for each object pair contained a separate join operation needs to be processed in the database. However, join operations are expensive, as the operation is quadratic in the number of objects (Böhm et al., 2000). Given a database table *DB*, a join operation requires projecting all rows of *DB* onto all rows of a copy of *DB*. Therefore, there is a reasonable motivation to develop methods to speed-up matching of QSQs against databases. Hence, this dissertation addresses the following research question:

*How can QSR and database indexing methods be applied to spatial databases in order to speed-up answering QSQs?*

In summary, there is a need for methods which connect the qualitative terms/concepts of a user and the quantitative data stored in the spatial database of GISs. To facilitate the intuitive and quick handling of QSQs in GISs, this dissertation aims at: (1) integrating qualitative spatial representations into GISs for enabling qualitative and intuitive formalism of queries in GISs and (2) developing novel database indexing and qualitative data reduction approaches for accelerating the interaction between GISs and their users.

## 1. INTRODUCTION

---

### 1.2 Application Scenarios

In this section, we present two kinds of application scenarios: (1) daily life and (2) disaster management.

#### 1.2.1 Daily Life

Very often visitors are interested in visiting a new city which has characteristics that differentiate it from other cities. In addition, visitors are looking for some places in the destination city that have some specific features. For example, they may want to find restaurants that are: *inside* a park and *near* a riverbank. Bremen is a historical city, which attracts many visitors from different countries every year. Hence, it would be beneficial to develop a system that enables visitors to quickly search for interesting places in Bremen using qualitative terms (e.g., they can formulate QSQs intuitively).

#### 1.2.2 Disaster Management

Every year, thousands of people are killed and hundreds of thousands more are displaced due to natural disasters such as earthquakes or floods. However, immediately after natural disasters such as the earthquakes in Sendai/Japan (2011) and Sulawesi/Indonesia (2012)<sup>1</sup>, fast response and recovery capabilities of Emergency Management Systems (EMS) play a crucial role in saving lives. Especially after large natural disasters, where precise data (in general from satellite images) is usually not available within the first 24 hours<sup>2</sup>. In addition, people have only a good chance to survive if they are rescued within 72 hours—the so-called “Golden 72 hours” (Jang et al., 2009). Thus, a major challenge to EMS is to respond to the QSQs of Emergency Managers (EMs) as fast as possible.

The ability to handle a QSQ such as “Find me: areas that are *near* the main street, but *far* away from damaged buildings, and *far* away from a riverbank” will help EMs to save the lives of many people (Al-Salman et al., 2013a). For instance, EMs can formulate queries to better direct rescue and aid distribution crews.

---

<sup>1</sup>See [www.emdat.be](http://www.emdat.be) for details

<sup>2</sup>According to RapidEye [www.rapideye.com](http://www.rapideye.com): satellite images can be available in 24-48 hours, (14.01.2014)

## 1.3 Dissertation Hypotheses and Contributions

Hypotheses:

- Spatial databases can be qualitatively, intuitively, and easily queried using qualitative descriptions.
- Qualitative spatial query processing is scalable in terms of space and time.

Based on these hypotheses, the main contributions of this dissertation are:

1. A theoretical framework that allows for integrating the appropriate qualitative spatial models into Spatial Data-Base Management Systems (SDBMSs). This framework enables the qualitative and intuitive formalism of queries in Geographic Information Systems (GISs).
2. The abstraction and administration of a Qualitative Spatial Layer (QSL) that covers the aspects of distance, topology, and direction in SDBMSs of GISs to enhance the processing time of QSQs.
3. Reducing the amount of qualitative data in the QSL by applying two strategies:
  - (a) Applying Density-Based Spatial Clustering of Applications with Noise (DB-SCAN) to group the database objects that are near to each other into clusters, and then identifying the inferable relations among clusters. Based on the identified inferable relations, we are able to avoid computing and storing some spatial relations in the QSL.
  - (b) Applying a converse operation to the qualitative models to exploit symmetry in the QSL, and thus reduce the size of the QSL.
4. Developing five optimization approaches to accelerate qualitative spatial query processing. These approaches combine QSR with database indexing approaches which are based on hash-tables and/or B<sup>+</sup>-trees data-structures or a combination of them.
5. **QualEnabler**, a practical system that combines the components of our work such as clustering and indexing.

## 1. INTRODUCTION

---

6. Innovative applications to enable intuitive and easy interactions with GISs.
7. Empirical studies using real-world and synthetic datasets to evaluate the proposed qualitative data reduction and indexing approaches.

### 1.4 Organization of the Dissertation

In this chapter, we have given a brief overview of the dissertation, then we have discussed the open problems and how this dissertation contributes to solving these problems. This dissertation is structured as follows:

Chapter 2 gives an overview of Qualitative Spatial representation and Reasoning (QSR) with a focus on the qualitative spatial calculi. Chapter 3 presents the state of the art in the Spatial Data-Base Management Systems (SDBMSs) and Geographic Information Systems (GISs).

Chapter 4 describes approaches for querying, reducing, and matching qualitative information. First, we integrate qualitative spatial models into SDBMSs. Then, based on the integrated models, we abstract the Qualitative Spatial Layer (QSL). Subsequently, we show how qualitative data reduction methods can be used to reduce the amounts of qualitative data in the QSL, since the space demands of the QSL are high.

Chapter 5 presents optimized indexing approaches for speeding-up answering QSQs in spatial databases. We show how  $B^+$ -trees and interpretation trees can be combined to form a new indexing approach. Afterwards, we combine qualitative spatial representations and hash-tables to develop hash-based indexing approaches.

In Chapter 6, we describe our practical system that we call **QualEnabler**. We elaborate on the implementation of the components of **QualEnabler** as well as its applications. Empirical studies carried out on real-world and synthetic datasets are reported in Chapter 7. Chapter 8 summarizes and discusses the results of this dissertation and gives future perspectives for SDBMSs and QSR research.

## Chapter 2

# Qualitative Spatial Representation and Reasoning

In everyday life, humans tend to rely on qualitative knowledge or abstractions rather than on measurements or prior quantitative knowledge to interact with the physical world and to reason about space and time (Cohn and Renz, 2008). For instance, we usually describe a person with the term *tall*, not by their precise value “172.9 cm”. We can additionally describe the order of people as *short* < *tall* in different situations without relying on any measurement. In spite of the great successes that have been achieved by systems and machines (e.g., super computers can do billions of calculations per second), they are not able to solve many real life problems as humans do. To do so, they should be able to rely on qualitative knowledge like humans (Cohn and Renz, 2008).

Dealing with common-sense knowledge qualitatively instead of quantitatively is an active research topic in several fields, including Geographic Information Systems (GISs), urban planning, and robot navigation (Wolter and Wallgrün, 2012). Qualitative reasoning, in turn, is a research area that aims to deal with common-sense knowledge using qualitative information. In order to cope with common-sense knowledge, one needs to first represent or abstract it using symbols or spatial relations. Qualitative reasoning particularly tries to deal with common-sense knowledge in a human-like manner. Additionally, it creates the possibility of coping with knowledge even when it is incomplete.

## 2. QUALITATIVE SPATIAL REPRESENTATION AND REASONING

---

Qualitative Spatial representation and Reasoning (QSR) is a sub-field of qualitative reasoning. It aims at developing and applying qualitative spatial calculi and their operations, that can be used to abstract knowledge as relations instead of measurements. For example, the Region Connection Calculus (RCC-8) (Cohn et al., 1997) is a binary qualitative topology calculus that can be used to abstract the topological relations of a given geometry. These relations are then used to reason about the space and time of common-sense knowledge (Cohn and Renz, 2008). QSR is valuable due to the fact that it allows for reasoning about space and time even if precise quantitative knowledge is not available, or if knowledge is incomplete.

In this chapter, we first describe qualitative spatial representations. The definitions of qualitative spatial calculi and their operations are then given in Section 2.1.1. Afterwards, the relevant aspects of qualitative spatial calculi are described in Section 2.2. Section 2.3 defines Qualitative Constraint Networks (QCNs). Conceptual neighborhood-based reasoning is described in Section 2.4. Finally, Section 2.5 sketches the consistency and relaxation methods of QCNs.

### 2.1 Qualitative Spatial Representations

Knowledge about spatial configurations or situations can be represented by identifying relationships between objects in space (Cohn and Renz, 2008). QSR assumes that qualitative representation or abstraction is done via a finite set of symbols  $\mathcal{B} = \{R_1, \dots, R_m\}$ , usually referred to as qualitative relations. Given a domain of interest  $\mathcal{D}$ , a  $k$ -ary qualitative relation defined over  $\mathcal{D}$  is a subset of the  $k$ -ary Cartesian product of the domain. In other words  $R \subseteq \mathcal{D}^k$ . The domain of interest in the scope of this dissertation is the set of simple regions<sup>1</sup> embedded in 2D space. Accordingly,  $\mathcal{D}^2$  is the set of any possible pairs of regions.

The relations in  $\mathcal{B}$  are usually referred to as base (or atomic, basic) relations. In general, they are Jointly Exhaustive and Pairwise Disjoint (JEPD), where  $\mathcal{B}$  covers all possible object pairs in the domain  $\bigcup_{R_i \in \mathcal{B}} R_i = \mathcal{D}^2$  (JE) and any domain object pair is contained in one, and only one base relation  $R_i \cap R_j = \emptyset \quad \forall R_i, R_j \in \mathcal{B} \text{ with } i \neq j$  (PD). Base relations suffice for providing a crisp description of a spatial scene and uncertainty

---

<sup>1</sup>A simple region is a connected and hole-free region with crisp boundaries. This is topologically equivalent to a closed disc (Egenhofer and Franzosa, 1995).



can be addressed by considering the union of possible base relations which can be held. In other words, we can consider the power-set  $2^{\mathcal{B}}$  of  $\mathcal{B}$ , i.e., the set of *disjunctive relations*. In turn, the universal relation  $U = \mathcal{D} \times \mathcal{D}$  is applied when no information is known.

### 2.1.1 Qualitative Spatial Calculi and their Operations

The definition of relations suffices for representation purposes. For reasons of completeness, we note that such a representation with a set of reasoning operations over the relations, among them standard set operations such as union and intersection, defines a so-called Qualitative Spatial Calculus (QSC). Based on such qualitative spatial calculi, spatial relations can be abstracted from given quantitative values.

A QSC is typically defined over a set of relations of uniform arity  $k$ , in which case one speaks of an  $k$ -ary calculus. In this dissertation, we will only consider binary calculi. If a relation  $R$  holds between two regions (domain objects)  $x$  and  $y$  then we say  $(x, y) \in R$  or simply  $xRy$ .

The relations in  $\mathcal{B}$  of a QSC are defined as standard sets; therefore they inherit set-theoretic operations such as union, intersection, and complement. For  $k$ -ary relations  $R$  and  $S \in 2^{\mathcal{B}}$ , union, intersection, and complement are formally defined as:

**union:**  $R \cup S = \{r \mid r \in R \vee r \in S\}$

**intersection:**  $R \cap S = \{r \mid r \in R \wedge r \in S\}$

**complement:**  $\overline{R} = \{r \mid r \in U \wedge r \notin R\}$

where  $r$  is a  $k$ -tuple and  $r \in D$ .

Additionally, a QSC provides a *converse* operation which is formally defined as:

**Definition 1** (Converse $^{\smile}$ ). *Given a binary relation  $R \in 2^{\mathcal{B}}$ , its converse operation is defined as:*

$$R^{\smile} = \{(x, y) \mid (y, x) \in R\}$$

Qualitative spatial calculi perform reasoning via a *composition* operation (see Definition 2) which is exploited from spatial relations.

**Definition 2** (Composition ( $R \circ S$ )). *Given two binary relations  $R$  and  $S \in 2^{\mathcal{B}}$ , their composition is defined as:*

$$R \circ S = \{(x, z) \mid (\exists y \in \mathcal{D}) : ((x, y) \in R \wedge (y, z) \in S)\}$$

## 2. QUALITATIVE SPATIAL REPRESENTATION AND REASONING

---

Nr	Type of relation	Relation
1	Direction	LEFT OF
2	Direction	RIGHT OF
3	Direction	BESIDE
4	Direction	ABOVE
5	Direction	BELOW
6	Direction	BEHIND
7	Direction	IN FRONT OF
8	Distance	NEAR
9	Distance	FAR
10	Topology	TOUCHING
11	Topology	BETWEEN
12	Topology	INSIDE
13	Topology	OUTSIDE

**Table 2.1:** Thirteen spatial relations from (Freeman, 1975).

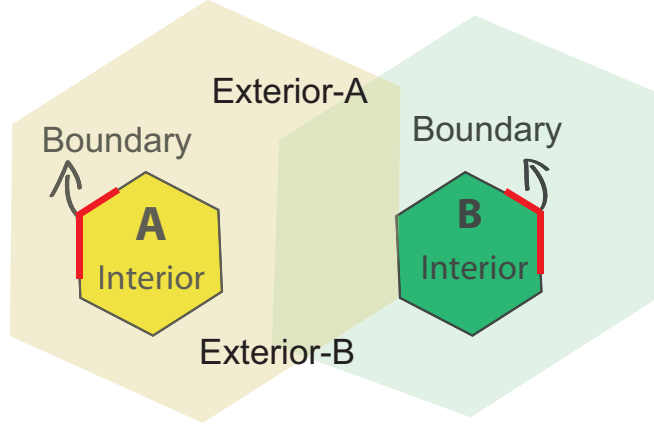
The composition of binary relations can be obtained from a precomputed lookup table called a *composition table*.

### 2.2 Aspects of Qualitative Spatial Calculi

During the last several decades, multiple categories of formal qualitative spatial calculi (e.g., directional calculi) for qualitative relations have been proposed. Freeman (1975) has proposed the thirteen spatial relations reported in Table 2.1, which are beneficial for developing several real life applications, including Geographic Information Systems (GISs). These relations cover the spatial aspects (or features) of topology (Section 2.2.1), distance (Section 2.2.3), and direction (Section 2.2.2). In addition, orientation (Section 2.2.2) has been shown to be an important aspect of space (Zimmermann and Freksa, 1996).

#### 2.2.1 Topology

The 9-Intersection Model (9IM) is proposed in (Egenhofer and Franzosa, 1995). It differentiates eight Jointly Exhaustive and Pairwise Disjoint (JEPD) relations between two simple convex regions without holes: *equal*, *disjoint*, *meets*, *overlaps*, *contains*, *covers*, *inside*, and *coveredBy*. Although the underlying definition of regions in the



**Figure 2.1:** Interior, boundary, and exterior of two regions  $A$  and  $B$ .




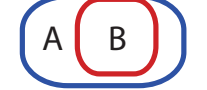
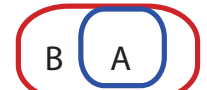
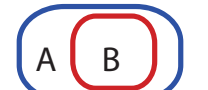
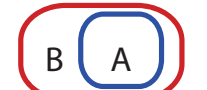
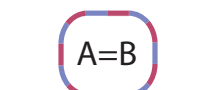
Region Connection Calculus (RCC-8) differs from the one in the 9IM, it uses the same eight base relations (Cohn et al., 1997). The 9IM differs from RCC-8 in considering interior, boundary, and exterior (or complement) point sets of every simple region. In particular, the binary relation between two regions  $A$  and  $B$  embedded in  $R^2$ , can be expressed based on the intersection of  $A$ 's interior ( $A^\circ$ ), boundary ( $\varphi A$ ), and exterior ( $A^-$ ) with  $B$ 's interior ( $B^\circ$ ), boundary ( $\varphi B$ ), and exterior ( $B^-$ ) (see Figure 2.1). Accordingly, the spatial relations between region pairs can be expressed by a  $3 \times 3$  matrix the so-called 9-intersection matrix (see Equation 1). Each intersection in the matrix returns 1 to indicate that two object parts intersect, or 0 if they do not. Therefore,  $2^9 = 512$  combinations can be generated. However, only eight of them are meaningful as spatial relations, since they are JEPD.

$$\textbf{Equation 1 (9IM). } 9IM(A,B) = \begin{Bmatrix} A^\circ \cap B^\circ & A^\circ \cap \varphi B & A^\circ \cap B^- \\ \varphi A \cap B^\circ & \varphi A \cap \varphi B & \varphi A \cap B^- \\ A^- \cap B^\circ & A^- \cap \varphi B & A^- \cap B^- \end{Bmatrix}$$

As depicted in Figure 2.2, the 9IM distinguishes the eight distinct topological relations from two regions embedded in  $\mathcal{D}^2$ .

**The Dimensionally Extended 9-Intersection Model (DE-9IM):** is proposed (Clementini et al., 1993) as an extension to the 9-Intersection Model, where the dimensions of intersected parts between two regions  $A$  and  $B$  are explicitly considered. For example, -1 value is given for the dimension of empty sets  $\emptyset$ , while non-empty sets  $\neg\emptyset$

## 2. QUALITATIVE SPATIAL REPRESENTATION AND REASONING

			
$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ Disjoint	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ Meet	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ Overlap	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ Covers
			
$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ CoverdBy	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ Contains	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ Inside	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Equal

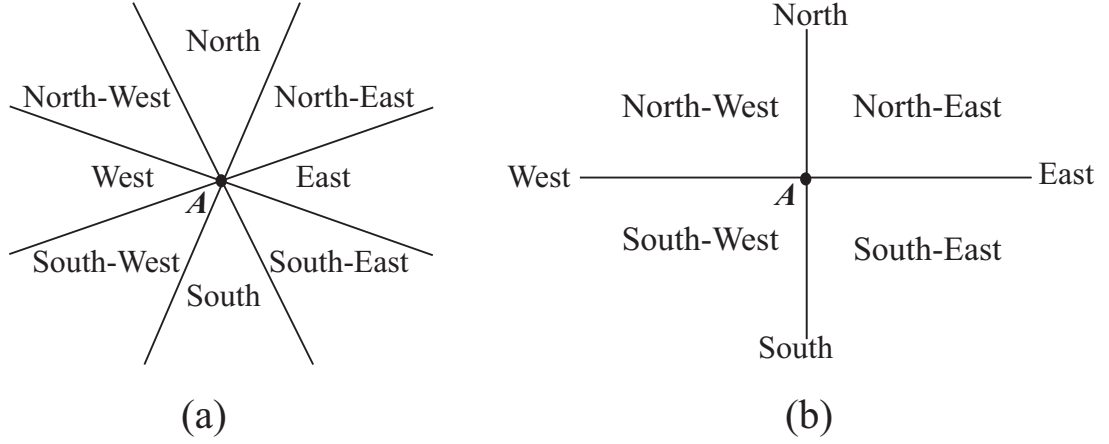
**Figure 2.2:** the eight distinct topological relations from two points sets  $(A, B)$  embedded in  $\mathcal{D}^2$  with their matrix values.

are given values; 0 for intersected points, 1 for intersected lines, and 2 for intersected areas. However, the simplest version of the DE-9IM maps the values of empty sets  $\emptyset$  (-1) to FALSE and non-empty sets  $\neg\emptyset$  (0,1, and 2) to TRUE.

### 2.2.2 Direction and Orientation

The direction relation between a pair of objects can be determined using three elements: a reference object, a primary object, and a Frame Of Reference (FOR) (Levinson, 1996). When the FOR depends on a fixed environment, it is called an extrinsic FOR. For example, the cardinal directions (*North*, *East*, *West*, and *South*) can be viewed as an extrinsic FOR to define a direction relation between a reference and a primary object. Furthermore, if an extrinsic FOR<sup>1</sup> is given, then the direction calculi can be expressed by the binary qualitative spatial relations. Given a reference object ( $A$ ) and a primary object ( $B$ ), the cardinal direction calculus differentiating nine cardinal directions: *South*,

<sup>1</sup>Sometimes in the literatures the extrinsic FOR is called an absolute FOR and accordingly a direction relation is called an absolute direction relation.



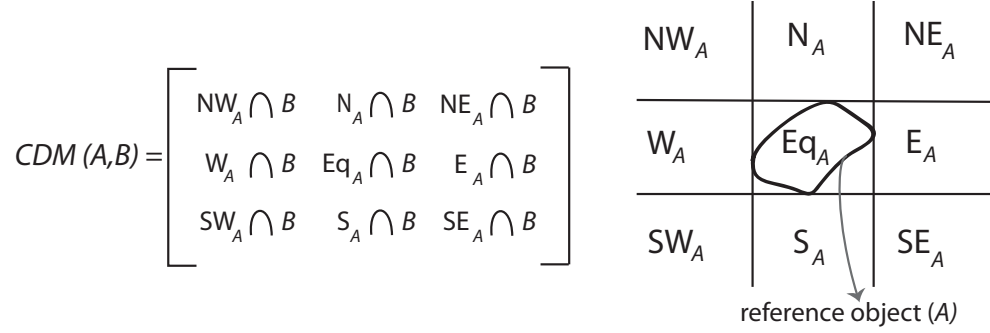
**Figure 2.3:** (a) the cone-based and (b) the projection-based models.

*South West*, *West*, *North West*, *North*, *North East*, *East*, *South East*, and *Equal* was introduced in (Frank, 1992).

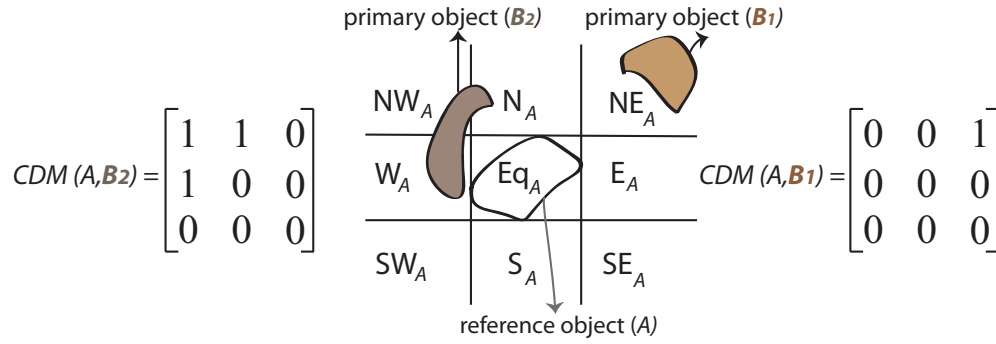
Frank (1992) particularly proposes two different partition schemes for point-based objects: (1) the projection-based model and (2) the cone-based model. Given the reference object ( $A$ ), the projection-based model uses horizontal (from  $W$  to  $E$ ) and vertical lines (from  $S$  to  $N$ ) crossing  $A$  to slice the space into eight directional relations (see Figure 2.3(b)). The cone-based model, in turn, slices the space around  $A$  into eight  $45^\circ$  partitions (see Figure 2.3(a)). The main difference between the aforementioned direction models is that the projection-based requires exact quantitative values to capture some directional relations (e.g.,  $N$ ), whereas cone-base is still able to capture any directional relation even when quantitative values are not exact.

The cardinal direction model, which can be used to abstract directional relations for extended objects, is proposed in (Skiadopoulos and Koubarakis, 2004). Based on the Minimum Bounding Rectangle (MBR) of the reference object ( $A$ ) the space is partitioned into nine regions connected to the nine directions in the Cardinal Direction Model (CDM). The primary object ( $B$ ) may be completely contained in one of the nine regions, called ‘single tile relation’. However, as the model deals with extended objects, a primary object may cover more than one region (partially or totally), which leads to 512 ‘multi-tile’ or conjunctive direction relations with respect to a reference object. In (Skiadopoulos and Koubarakis, 2004) 218 consistent and JEPD relations out

## 2. QUALITATIVE SPATIAL REPRESENTATION AND REASONING



**Figure 2.4:** A 3x3 matrix is used to represent the binary directional relation(s) between  $A$  and  $B$ .



**Figure 2.5:**  $B_1 \{[NW]\}A$  and  $B_2 \{[W, NW, N]\}A$ .

of 512 possible direction relations are distinguished. In the CDM, the binary directional relation(s) between  $A$  and  $B$  can be represented using a  $(3 \times 3)$  matrix (see Figure 2.4).

As depicted in Figure 2.5,  $B_1$  is in the (single) relation  $NE$  with  $A$ , and  $B_2$  is in a conjunctive relation  $[W, NW, N]$  with  $A$ .

Orientation calculi are developed based on intrinsic FOR, where an orientation relation can be determined based on the properties of a primary object or a reference object (e.g., an intrinsic front of an object). For example, the Oriented Point Relation Algebra ( $OPRA_m$ ) (Dylla and Moratz, 2004; Moratz and Ragni, 2008) correlates the pairs of points with each other based on their relative orientation and location in a 2D-space.

### 2.2.3 Distance

Distance is an important and complex aspect of space. Gahegan (1995) points out that the notion of distance is context-dependent, and humans perception of distances can be mainly influenced by three factors: (1) the effect of scale, (2) the “attractiveness” of objects, and (3) the effect of reachability. For instance, what might be regarded as close at one scale, could be called far away at another scale. For example, the following sentences are not contradictory:

1. *Bremen is near Hanover, but far from Dubai.*
2. *Bremen is near Dubai, but far from the Moon.*

The “attractiveness” of objects is also an important factor regarding our perception of distances. For example, being near to the shopping center of Bremen is not necessarily interpreted the same by all people.

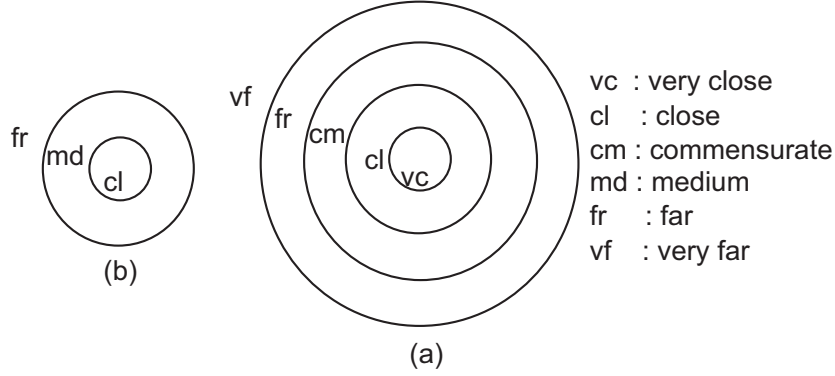
Reachability is a crucial factor as well. For example, the street network of a city might constrain the movements between objects by different means of transportation (e.g., train, car, or bike).

Distance calculi can be divided into two categories: absolute and relative. In the absolute distance calculi, distance relations can be derived based on the used absolute scale of space. In such calculi, distance relations can be abstracted with a linear Euclidean distance, where the maximum line of Euclidean distance is divided into several intervals, such as *near* or *far*.

In turn, the relative distance calculi can be used to abstract the distance relations by comparing the relative distance to a given reference distance. For example, Hernández et al. (1995) proposes a distance model that can be used to abstract distance relations based on distance intervals and three elements: a reference object, a primary object, and the Frame Of Reference (FOR). Three operators  $>, =, <$  can be used for comparing distances. In the distance model, a reference object splits its surrounding space into a number of ordered distance relations  $Q = \{q_0, \dots, q_n\}$ , where  $q_0$  is the shortest distance to a reference object and  $q_n$  is the longest. The acceptance areas of distance relations are represented as the monotonic increasing intervals  $\delta_n$ :  $(\delta_0 \leq \delta_1 \leq \dots \leq \delta_{n-1} \leq \delta_n)$ . In order to deal with uncertainty, the distance model may have different levels of granularity, which allows one to move to the next coarser level when no decision can be

## 2. QUALITATIVE SPATIAL REPRESENTATION AND REASONING

---



**Figure 2.6:** Two levels of granularity of distance model, adapted from (Hernández et al., 1995).

made at the present level. Figure 2.6 shows the two variants of the distance model, one on a coarse level (Figure 2.6(b)), and the other on a finer level (Figure 2.6(a)).

### 2.2.4 Other Aspects of Qualitative Spatial Calculi

Aside from the aforementioned calculi, other qualitative calculi that combine two calculi are proposed. For example, the Ternary Point Configuration Calculus (TPCC) that models the relative position between two objects as points (a reference object and a primary object) is presented in (Moratz et al., 2003). In this approach, the qualitative distance and direction relations are combined to represent the relative position (e.g., front-close) of a primary object with respect to a reference object.

## 2.3 Qualitative Constraint Networks

Spatial knowledge about qualitative spatial relations and objects can be given in the form of constraints. Generally speaking, such constraints can be formalized as a Constraint Satisfaction Problem (CSP), where a CSP contains a set of domain objects (or variables)  $O$  and  $k$ -ary spatial relations ( $k$ -ary constraints). The CSP can be formalized as a labelled graph: the-so-called Qualitative Constraint Network (QCN),  $QCN=(O, C)$ , where  $O$  is a set of domain objects, and  $C$  is a set of constraints over  $O$ . A formal definition of a QCN is given as follows:



**Definition 3** (Qualitative Constraint Network). A Qualitative Constraint Network (QCN) over a qualitative calculus  $\zeta$  is a pair  $(O, C)$  where:

- $O = \{o_1, \dots, o_n\}$  is a set of domain objects,
- $C : O \times O \rightarrow R_\zeta$  is a function mapping each pair of objects from  $O$  to a relation of  $\zeta$ , where  $C(o_i, o_j) = R_{ij} \in R_\zeta$  such that relation  $R_{ij}$  has to hold for the values assigned to  $o_i$  and  $o_j$ ,
- for all  $i \geq 1, j \leq n$ ,  $C(o_i, o_i) = id$  and  $C(o_i, o_j) = C(o_j, o_i)^\smile$ , where  $id$  is the identity relation of  $\zeta$ .

Representing a set of objects and their spatial relations as a QCN allows for applying qualitative spatial reasoning operations to several applications. Such techniques can be used to forward-prune the search space of spatial databases, thus speeding-up query answering. In addition, such representation allows methods such as algebraic closure (Mackworth, 1977) to check the consistency of QCNs.

## 2.4 Conceptual Neighbourhood-Based Reasoning

Freksa (1991, 1992) proposes a conceptual neighborhood-based reasoning approach that is based on Allen’s interval algebra (Allen, 1983). According to Freksa (1991), “Two relations between pairs of events are conceptual neighbors, if they can be directly transformed into one another by continuously deforming (i.e., shortening, lengthening, moving) the events in a topological sense”. For instance, assume we have two cars  $X$  and  $Y$  in a race.

Then we can distinguish several temporal relations between  $X$  and  $Y$  during the race, such as at times  $t_0$  ( $X < Y$ ),  $t_2$  ( $X = Y$ ), and  $t_3$  ( $X > Y$ ). In this case, the relations ( $X < Y$ ) and ( $X = Y$ ) are conceptually neighbored because it is possible to directly and continuously move from  $t_0$  to  $t_1$ . However, since it is not possible to continuously move from  $t_0$  to  $t_2$ , ( $X < Y$ ) and ( $X > Y$ ) are not conceptually neighbored. The concept of events-neighborhood has been adapted to spatial entities by (Dylla and Wallgrün, 2007), where the authors represent the spatial relations of qualitative spatial calculi as Conceptual Neighborhood Graphs (CNGs).

## 2. QUALITATIVE SPATIAL REPRESENTATION AND REASONING

---

### 2.5 Consistency and Relaxations

Due to imprecise data coming from sensors, qualitative spatial relations which are abstracted from sensor data may lead to inconsistent QCNs. In this case, one needs to apply algorithms such as algebraic closure (van Beek, 1992) to check the consistency of QCNs. Two approaches are presented in (Egenhofer, 1994a; Wallgrün et al., 2010), and use algebraic closure to check whether the constraints of QCNs of a spatial query are free of contradiction (or conflict). One of the approaches to cope with inconsistency is to relax the constraints of QCNs. For example, Dylla and Wallgrün (2007) propose a relaxation function to obtain coarse relations from the base relations of original QCNs. This function is based on a conceptual neighborhood structure of QCNs called Conceptual Neighborhood Graphs (CNGs). Accordingly, a distance function that finds the minimal relaxations of QCNs based on CNGs is proposed. Such an approach could be very helpful in real-life applications, such as, in GIS, where spatial queries could be inconsistent, and thus the relaxation can be applied to find the conceptually neighboring spatial queries.

### 2.6 Summary

In the context of Qualitative Spatial representation and Reasoning (QSR), we have first described qualitative spatial representations, qualitative spatial calculi, and their operations in Section 2.1. We have focused on reviewing the aspects of qualitative spatial calculi that are more related to this dissertation. We have mainly explained the topology, distance, and direction calculi in Section 2.2. Qualitative constraint networks have been elaborated in Section 2.3. Lastly, we have described conceptual neighborhood-based reasoning (Section 2.4) and consistency and relaxations (Section 2.5).

# Spatial Information Management and Systems

“Geographic Information Systems (GISs) are computer-based information systems that are used to capture, model, store, retrieve, share, manipulate, analyze, and present geographically referenced data” (Worboys and Duckham, 2004). GISs usually rely on spatial database management systems to manage huge amounts of spatial data which are stored in spatial databases (see Section 3.1).

In the context of Qualitative Spatial representation and Reasoning (QSR), clustering techniques are typically used with spatial database to reduce qualitative spatial information that are stored within them (see Section 3.2).

In order to enable an intuitive and efficient interaction between GISs and their users, the techniques of QSR are integrated with GISs (see Section 3.3).

## 3.1 Spatial Data-Base Management Systems

At the heart of every geographic information system, Spatial Data-Base Management Systems (SDBMSs) are usually used to manage and retrieve huge amounts of geographical information which are stored in spatial databases. In order to enable querying the spatial databases, queries need to be supported by SDBMSs (see Section 3.1.1). SDBMSs do not only use appropriate data-structures and spatial data types for storing spatial data efficiently, but they also apply indexing methods (see Section 3.1.2) to efficiently handle spatial queries.

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

#### 3.1.1 Spatial Queries

Traditional Data-Base Management Systems (DBMSs) are typically developed to handle non-spatial queries such as: “In the employee database table, list the *names* of all employees who have a *salary* >10000 dollar”. To answer this query, DBMSs use relational algebra operators such as projection (e.g., salary >10000). To accelerate processing such queries, DBMSs usually apply B-trees and hash-based indexing on the numeric and non-numeric attributes of database tables.

However, in order to be able to process spatial queries such as, “Find all cities in North Rhine-Westphalia”, DBMSs must have three additional elements: (1) a spatial data model, (2) a spatial query language (e.g., SQL3), and (3) a spatial indexing method such as R-trees, which are usually applied to the spatial data types attributed to database tables. The three elements can be viewed as a spatial extension to DBMSs which inherits algebraic set operations (e.g., union or intersection). Furthermore, the spatial extension does not change the functionality of the original DBMS. In this case, DBMSs can be called Spatial DBMSs (SDBMSs).

##### 3.1.1.1 Spatial Data Models

A spatial data model aims to give a high level description of spatial data. Güting (1994) points out that a successful spatial data model should meet three properties: (1) Spatial Data Types (SDTs) (e.g., lines, points or polygons) should store spatial data or geometric entities, (2) be simple data structures, and (3) be spatial operators (e.g., *inside*(park, Bremen), the whole part of a park is *inside* Bremen) that can be performed over SDTs.

Several approaches have been proposed to model and manage spatial data, and they can be divided into two categories (Manolopoulos et al., 2005): (1) GIS-centric and (2) DBMS-centric. In the first category, the approaches are proposed to deal with raster data or map layers. In (Johnston and Redlands, 2004; Tomlin, 2012), a map is represented as 2D grids or cells, where each grid has a property (e.g., temperature). In order to have more than one property for the grids of a map, the map needs to be duplicated into several map layers, where the number of layers equals the number of properties, and the grids of each layer have a distinct property. Additionally, a set

of spatial operations such as fusion, can be applied on a map which will lead to the generation of a new map.

In the DBMS-centric category, many approaches have been proposed to handle vector-based data (e.g., geometries). Güting and Schneider (1995) propose the RObust Spatial Extension (ROSE) algebra, a general independent data model which provides abstractions for points, lines, and regions. ROSE provides spatial operations over two sets ( $GEO = \{\text{points, lines, regions}\}$ ,  $EXT = \{\text{lines, regions}\}$ ). Traditional algebraic operations (e.g., intersection) as well as some topological relations (e.g., *inside*), can be performed over the sets.

Egenhofer and Franzosa (1991) propose the four-intersection model that can be used to compute the binary topological relations among the pairs of simple regions (without holes). In this model, the interiors and boundaries of the interested region pairs are used to capture the topological relations. Later, the model was extended to the nine-intersection model (Egenhofer and Franzosa, 1995), in which the interiors, boundaries, and exteriors were used to capture the topological relations of the intersected regions. Some other relational approaches, such as (Güting, 1988), consider storing spatial data in relations, where the data is stored as atomic values, and satisfies the First Normal Form (1NF). However, the mentioned works did not address the spatial data modeling issues.

In order to address the spatial data modeling issues, object-oriented models have been proposed (Cheng and Gadia, 1994; Clementini and Di Felice, 1993; Günther and Riekert, 1993). The models satisfy the 1NF and inherit the properties and capabilities of object-oriented models, in which spatial data types, data structures, and spatial operators are encapsulated in a class.

The Geometry Object Model (GOM)<sup>1</sup> has also been developed (International Organization for Standardization (ISO) (ISO/IEC, 2002) and (Open Geospatial Consortium (OGC) Inc., 2011)) as a standard object relational model.

The model is already used by several spatial database vendors (e.g., ORACLE<sup>2</sup>) and open source suppliers (e.g., PostgreSQL<sup>3</sup>). It provides spatial operators for most spatial data types and supports R-tree spatial indexing. As shown in Figure 3.1, the model is

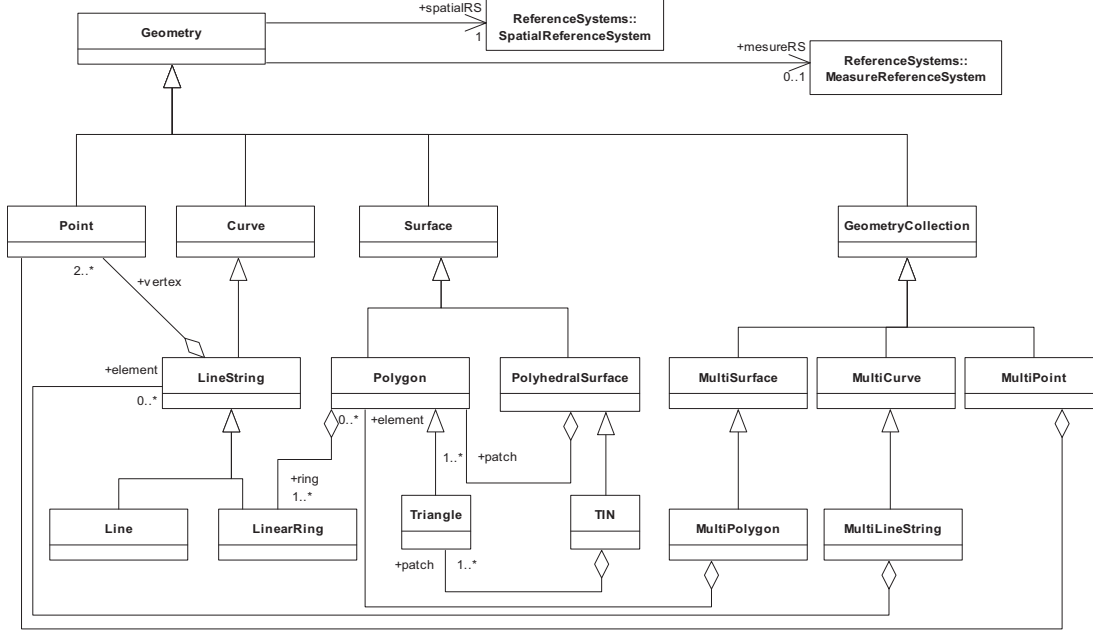
---

<sup>1</sup>In the literature it is also called OGIS spatial data model.

<sup>2</sup>ORACLE: [www.oracle.com/](http://www.oracle.com/)

<sup>3</sup>PostgreSQL: [www.postgresql.org/](http://www.postgresql.org/)

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS



**Figure 3.1:** The geometry object model, from (Open Geospatial Consortium (OGC) Inc., 2011).

described by the Unified Model Language (UML) and has a super Geometry class with subclasses: Point, Curve, Surface, and GeometryCollection. The model also describes the relationships between the subclasses, and incorporates a spatial reference system to determine the position (space coordinate) of each geometric object. Aside from the traditional algebraic set operators, the GOM provides a rich set of spatial operators over geometric objects. Table 3.1 lists three categories of spatial operations: (1) basic operators, (2) topological set operators, and (3) spatial analysis operators. The first category contains general and unary operations to get information (e.g., features) about a geometric object. The second category represents the eight topological predicates that are developed based on the Dimensionally Extended 9-Intersection Model (DE-9IM) (Clementini et al., 1993). The last category contains the algebraic set, distance-based, and convex hull operators. In this category, convex hull and buffer are unary operators.

In this dissertation, we will use the GOM, as it is a stable and extendable model.

### 3.1 Spatial Data-Base Management Systems

**Table 3.1:** Three kinds of spatial operations are provided: (1) basic operators, (2) topological set operators, and (3) spatial analysis operators, from (Open Geospatial Consortium (OGC) Inc., 2011).

<b>Basic Operators</b>	
SRID ( )	Returns the Spatial Reference System ID for this geometric object
Envelope ( )	The minimum bounding box for this Geometry, returned as a Geometry
IsEmpty ( )	Returns true if this geometric object is the empty Geometry
IsSimple ( )	Returns true if this geometric object has no anomalous geometric points
IsMeasured ( )	Returns true if this geometric object has m coordinate values
Boundary ( )	Returns the closure of the combinatorial boundary of this geometric object
<b>Topological/ Set Operators</b>	
Equals	Returns true if this geometric object is spatially equal to another Geometry
Disjoint	Returns true if this geometric object is spatially disjoint from another Geometry
Intersects	Returns true if this geometric object spatially intersects another Geometry
Touches	Returns true if this geometric object spatially touches another Geometry
Crosses	Returns true if this geometric object spatially crosses another Geometry
Within	Returns true if this geometric object is spatially within another Geometry
Contains	Returns true if this geometric object spatially contains another Geometry
Overlaps	Returns true if this geometric object spatially overlaps another Geometry
<b>Spatial Analysis Operators</b>	
Distance	Returns the shortest distance between any two Points in the two geometric objects as calculated in the spatial reference system of this geometric object
Buffer	Returns a geometric object that represents all Points whose distance from this geometric object is less than or equal to distance
ConvexHull	Returns a geometric object that represents the convex hull of this geometric object
Intersection	Returns a geometric object that represents the Point set intersection of this geometric object with another Geometry
Union	Returns a geometric object that represents the Point set union of this geometric object with another Geometry
Difference	Returns a geometric object that represents the Point set difference of this geometric object with another Geometry
SymDifference	Returns a geometric object that represents the Point set symmetric difference of this geometric object with another Geometry

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

#### 3.1.1.2 Spatial Query Languages

A Structured Query Language (SQL) is a high-level declarative language that allows its end-users to query DBMSs without knowledge about how to execute and optimize a query. SQL is based on relation set algebra operators, including select, project, union, cross-product, difference, and intersection. Additionally, SQL provides three query languages: (1) a data definition language (e.g., for creating a relation of schema), (2) a data manipulation language (e.g., for inserting rows into a database table), and (3) a data control language (e.g., for granting permissions on relations of database schema). SQL was designed to support conventional DBMSs but not spatial ones. Hence, SQL needs to be extended in order to support spatial data.

Several approaches have been proposed to extend the capabilities of SQL so that it can deal with spatial data in spatial databases. For example, Query-By-Example (Zloof, 1977) and Query-By-Pictorial-Example (Chang and Fu, 1980) use SQL, where spatial data such as the centroids of geometric objects are stored as floating point values while other spatial data is stored as strings. However, in these approaches, the users are required to be completely aware of the implementations of spatial data. Additionally, several spatial operations such as the topological operations *inside* and *overlap* cannot be performed since they require representing and storing the geometries (or at least their approximations such as the MBRs) of spatial objects.

In (Aref and Samet, 1991; Roussopoulos and Leifker, 1985; Samet and Aref, 1995), SQL has been extended to perform spatial operations and manipulate spatial data in spatial databases. In the proposed extensions, the structure of SQL is preserved, with the addition of spatial data types and operations. In (Egenhofer, 1994b), a spatial SQL has been proposed as a comprehensive spatial extension of SQL. In particular, the spatial SQL preserves the original SQL structure and concepts, incorporates spatial relations and operations, and involves spatial data types. The spatial SQL consists of two languages: (1) a spatial query language that allows users to submit spatial queries to retrieve spatial information and (2) a presentation language that allows users to specify how to display the retrieved spatial information (e.g., presenting results on a map).

Indeed there is a direct connection between spatial data models and the spatial query languages. Without a complete and practical spatial data model it is not possible



to provide an intuitive and efficient spatial query language. Thus, the Geometry Object Model (GOM) has been extended to SQL and termed OGIS/SQL (Open Geospatial Consortium (OGC) Inc., 2010), where the users can define spatial data types and perform spatial operations. Again, OGIS/SQL preserves the structure and the capabilities of the original SQL. Therefore, spatial, non-spatial queries, or a combination of them can be performed using OGIS/SQL. Güting (1994) and Rigaux et al. (2002) list out several kinds of spatial queries that can be performed by OGIS/SQL. Table 3.2 shows the types of spatial queries, spatial operations, spatial queries, and spatial indexing methods that can be applied to the spatial data types.

In this dissertation, we will use OGISs/SQL due to its capability of providing a rich set of functionalities to interact with SDBMSs.

#### 3.1.2 Spatial Indexing

Indexing is a data-structure designed to accelerate the retrieval of (spatial) data in (S)DBMSs. For instance, spatial indexing aims at speeding-up the retrieval of spatial data. In this section, three types of commonly used indexing in (S)DBMSs are described: B-trees (see Section 3.1.2.1), R-trees (see Section 3.1.2.2), and Hashing (see Section 3.1.2.3). Afterwards, the applications of indexing for spatial databases are explained (see Section 3.1.3).

##### 3.1.2.1 B-trees

B-trees are widely and commonly used data structures that aim to provide direct access methods on secondary storage devices (Comer, 1979). The main idea of a B-tree is to keep entities sorted in a balanced tree structure, even when the number of indexed entities grows and shrinks.

A B-tree is a balanced search tree and its worst case height is  $O(\log n)$ . As with any tree data structure, B-tree of order  $P$  has three types of nodes: root, leaf, and internal. Every internal node, has between  $M = P - 1$  and  $m$  children, where  $M$  is the maximum number of nodes and  $m \leq \lceil M/2 \rceil$  is the minimum number of nodes (Comer, 1979). Furthermore, every internal node is in the form of:

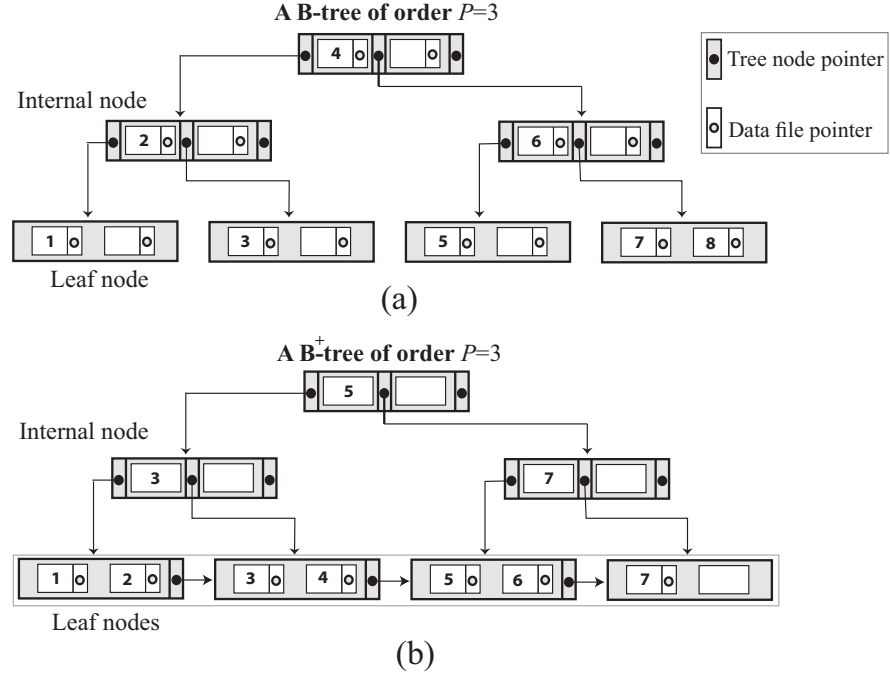
$\{Ptr_1, \langle K_1, Pd_1 \rangle, Ptr_2, \langle K_2, Pd_2 \rangle, \dots, \langle K_{z-1}, Pd_{z-1} \rangle, Ptr_z\}$ , where  $K_i$  is a key,  $Ptr_i$  is a tree (or child) pointer,  $Pd_i$  is a data file pointer, and  $z$  is the number of stored keys in the node. In particular, each internal node (Comer, 1979):

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

**Table 3.2:** The types of spatial queries, the possible spatial operations, spatial queries, and their spatial indexing methods.

Type of Query	Operation/Predicate	Query	Indexing
Containment	Contains or Covers	Find each object in the map that completely contains a search object $O$	R-trees
Region/Window	Intersects	Find each object in the map that intersects a search object $O$ or window $W$	R-trees
Line Intersection	Intersects	Find each object in the map that intersects a search line $L$	R-trees
Enclosure	CoveredBy or Inside	Find all objects in the map that are contained by a given object $O$ or window $W$	R-trees
Clipping	Intersects or CoveredBy or Inside	Extract all the portions of objects in the map that are covered by, inside or intersect a search object $O$ or window $W$	R-trees
Spatial Join ( $G, U$ )	Topological predicates such as Intersects	Given two sets of geometric objects $G$ and $U$ , find object pairs from $G$ and $U$ so that they satisfy a join predicate(s)	B-trees and/or R-trees and/or Hashing
Adjacency	Meets	Find all objects that are adjacent to a search object $O$ or window $W$	R-trees
Metric/Spatial Range	Distance	Find the minimum distance between objects $O$ and $O'$	R-trees
Nearest Neighbor	Distance and Buffer	Find the closest objects to a search object $O$	R-trees
Merge	Union	Return the geometric union of two objects $O$ and $O$	R-trees



**Figure 3.2:** Example of B-tree and B<sup>+</sup>-tree of order  $P = 3$ . The values are inserted in the order  $\{1, 2, 3, 4, 5, 6, 7, 8\}$ .

1. keeps keys strictly in ascending order ( $K_1 < K_2, \dots, K_{z-1} < K_z$ ).
2. has a key  $K_i$  associated with a pointer  $Pd_i$  to the data file block that contains the key.
3. has a key  $K_i$  associated with leftmost  $Ptr_i$  and rightmost  $Ptr_{i+1}$  tree (children) pointers, which implies that each node (except leaf nodes) contains  $z + 1$  pointers to children nodes.
4. has a  $Ptr_i$  ranges that include all the keys of its corresponding subtree that are less than or equal to  $K_i$ . In contrast,  $Ptr_{i+1}$  ranges include all the keys of its corresponding sub-tree that are greater than  $K_i$ .

In turn, each leaf node is in the form  $\{<K_1, Pd_1>, \dots, <K_{z-1}, Pd_{z-1}>\}$ . From the form of the leaf nodes, we can recognize that they are the same as internal nodes, but they do not have pointers to children nodes. In addition, all of the leaf nodes are located at the same level. The root node, on the other hand may contain one or more keys, but

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

always have pointers to children nodes. In summary, the keys of the successor nodes are always ordered by the keys of the predecessor nodes, which leads to logarithmic times search, insert, and delete. A B-tree of order  $P = 3$  is depicted in Figure 3.2(a).

*Search:* in order to find a specific key, the search operation of a B-tree traverses the keys of each internal node. It uses a top-down paradigm from the root of the tree arriving to the leaves. However, the keys are ordered and there is usually a possibility for pruning some sub-tree branches.

*Insert:* B-tree must be kept balanced. Hence, when a new key is inserted into the data file, the insert operation of a B-tree first checks the location of its index. Then it searches for an empty space in the B-tree to place the key. If a free space is found, then the key is simply inserted and there is no need to recursively reconstruct a B-tree (or sub-branches of it). However, if a new key causes an overflow in an internal node, then the node is partitioned equally into two nodes by a median (or pivot) key. Afterwards, a new internal node that contains the median key, and points to the two sets of partitioned nodes is created.

*Delete:* the delete operation is very similar to the insert function. After deleting a key from the index of a B-tree, the number of keys of affected internal nodes are checked, and finally a merge operation takes place if  $z < m$ . Otherwise, no action is taken.

**B<sup>+</sup>-tree:** is the most common variant of a B-tree, and is implemented in most current SDBMSs (e.g., PostgreSQL<sup>1</sup>). B<sup>+</sup>-trees differ from B-trees in that, (1) each internal node contains keys and tree pointers, but no data file pointers and (2) in addition to a tree pointer to the next leaf node, leaf nodes of the tree store all the keys associated with their data file pointers. Therefore, the structure of internal and leaf nodes of B<sup>+</sup>-trees differs from the ones of B-trees. In a B<sup>+</sup>-tree, each internal node is in the form of  $\{Ptr_1, K_1, Ptr_2, \dots, Ptr_{z-1}, K_{z-1}, Ptr_z\}$ . Internal nodes do not need to store data file pointers and they are capable of packing more entries. Hence, B<sup>+</sup>-trees can have lower levels than B-trees, leading faster searches of B<sup>+</sup> trees. Each leaf node is in the form

$\{<K_1, Pd_1>, <K_2, Pd_2>, \dots, <K_{z-1}, Pd_{z-1}>, Ptr_{NEXT}\}$ . In addition, each leaf node contains a tree pointer to the next leaf node, which allows traversing leaf nodes as a linked-list. Moreover, some keys of internal nodes are duplicated and stored in leaf nodes to guide the search. A B<sup>+</sup>-tree of order  $P = 3$  is depicted in Figure 3.2(b). The

---

<sup>1</sup>PostgreSQL: <http://www.postgresql.org/>

search, insert, and delete operations of  $B^+$ -trees are quite similar to the ones of B-trees, although the keys and their duplications are arranged differently.

We will use  $B^+$ -trees indexing in this dissertation due to their abilities of faster searches.

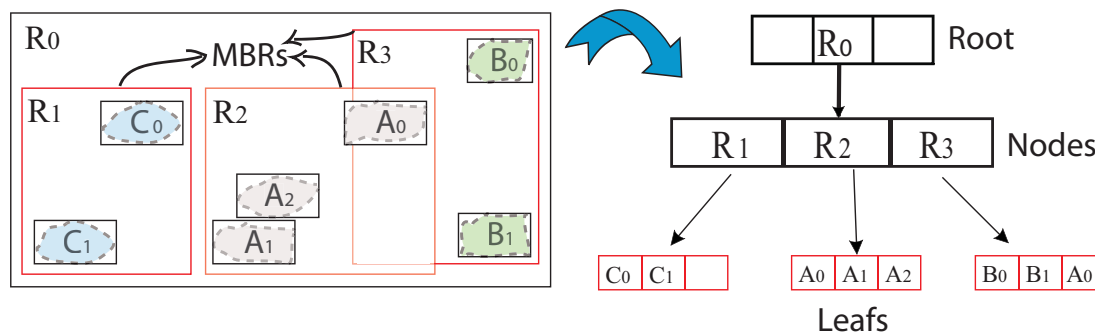
#### 3.1.2.2 R-trees

R-trees (Guttman, 1983) have been proposed as an extension to B-trees to support multi-dimensional data. R-trees support spatial access methods by indexing multi-dimensional data (e.g., polygons or geo-coordinates), and are commonly implemented in spatial databases of GISs. In general, constructing R-tree indices results in computing the Minimum Bounding Rectangles (MBRs) of objects, which are then clustered into groups in the next higher level of the tree, where the MBRs of all the objects contained are calculated.

Formally, the leaves of R-trees reside on the same level and are structured as pairs  $(ML, OL)$ , where  $ML$  is the MBR of a spatial object  $OL$ . In the next higher level, the parent or internal nodes of R-trees are formulated as pairs  $(MC, CP)$ , where  $MC$  is the MBR which contains all the MBRs of the children and  $CP$  is a child pointer. For instance,  $R_0$  in Figure 3.3 is a root node and points to the MBRs of three nodes  $R_1, R_2$ , and  $R_3$ , while these nodes point to the MBRs of leaf nodes. Moreover, any parent node must have between  $m$  and  $M$  children, where  $M$  is the maximum number of nodes (or objects in the leaf-level) per MBR and  $m \leq \lceil M/2 \rceil$  is the minimum number of nodes per MBR. In R-trees, splitting the space into MBRs is a critical and crucial method. Particularly, the *split* function attempts to divide the space into MBRs to minimize the overlapping between MBRs, so that the bounding rectangle of the whole space can be minimized as much as possible. In order to guarantee the optimal division of space, the split function would have to check every pair of objects, which requires an exponential number of steps. Hence suboptimal linear and quadratic variants of the split function are often used to split the space in a feasible time-frame.

R-trees provide efficient search, insert, and delete operations, which allows one to easily update and integrate incoming data into the database. Search and insert operations are described as follows:

*Search* : similar to B-trees, R-trees start at the root of the index ( $T$ ) and traverse all of the related nodes, up to leaves, in logarithmic time. Given a spatial query  $Q$  (e.g.,



**Figure 3.3:** An example of an R-tree for 2D geometric objects.

a region query), the search operation aims at finding all the MBRs of leaves that intersect  $Q$ . If a node of  $T$  is an internal node, R-trees search every  $CP$  of  $MC$  that overlaps  $Q$ . Once the leaves are reached, every  $OL$  is retrieved if its  $ML$  intersects  $Q$ .

*Insert:* the insertion (and similarly deletion) of new object  $NewObj$  requires locating a leaf starting from  $T$  to insert  $NewObj$ . If there is a leaf that has an empty space,  $NewObj$  is inserted into it. Otherwise, a split operation takes place and all MBRs changes are propagated recursively (upwards-manner).

**R<sup>+</sup>-trees:** are proposed as a modified version of R-trees (Sellis et al., 1987). Unlike R-trees, R<sup>+</sup>-trees prevent intersecting the MBRs at the same level of a tree that may require including an inserted object<sup>1</sup> in more than one MBR. In other words, although each internal node can only be visited once, the same objects can be redundant in different internal nodes, which may lead to an increase in the size and number of nodes.

**R\*-trees:** are also proposed as a modified and very well received and accepted version of R-trees (Beckmann et al., 1990). R\*-trees have the advantage that they have no limitation on the number of nodes. In addition, they attempt to minimize the unused spaces between the MBRs by reinserting objects in the appropriate MBRs. R\*-trees also attempt to reduce the intersections between MBRs.

#### 3.1.2.3 Hashing

Hash tables have successfully been adapted and applied to databases to allow direct access to stored data. In databases, hash tables are known as hash files. These hash files map records (tuples) of a data file of database into buckets. Each bucket is represented

<sup>1</sup>An object is identified by a unique id.

as a linked-list that contains a number of records. If a hash file has  $M$  buckets, hashing is done by a hash function  $h$  that converts the value of a record  $r$  into a fixed-size number to map  $r$  to the bucket location of a hash file. One common and simple hash function is  $h(r) = K \bmod M$ . However, since the number of possible hash keys is usually much larger than  $M$ , several keys could be mapped to the same bucket, which leads to a so-called collision. Collisions in databases can be handled by simply inserting  $r$  to the first empty record in the bucket. When the bucket is full, new mapped keys to the bucket cause overflow and can be handled by chaining, in which case the overflowed keys are inserted into a new bucket that is connected via a pointer to the old one. This type of hashing is known as a static hashing because it is bounded by a fixed number of  $M$  buckets.

The other common type of hashing is called dynamic hashing. Linear hashing is one of the most popular dynamic hashing methods, in which the size of hash file is able to dynamically grow and shrink with its database. Linear hashing starts to map records into buckets using  $h(r) = K \bmod M$ . When a bucket records an overflow, the size of the hash file is doubled to  $2M$ , and the records of the hash file are distributed through  $2M$  buckets using the new hashing function  $h'(r) = K \bmod 2M$ . Therefore, linear hashing is suitable for databases that contain a large amounts of data.

Accordingly we will use and apply linear hashing throughout, this dissertation.

#### 3.1.3 Indexing Applications for Spatial Databases

Recently, several indexing approaches have been proposed to cope with spatial query processing. In this section, we focus on the proposed hash-based indexing approaches to handle spatial queries, since hashing is the main focus of this dissertation.

##### **B-trees:**

$B^+$ -tree indexing that uses space-filling curves such as the Peano curve (or Z-curve) and the Hilbert curve (Faloutsos and Roseman, 1989) to handle spatial queries are presented in (Faloutsos and Rong, 1991; Jensen et al., 2004b; Yiu et al., 2008). Given objects in 2D-space, the representation of their locations can be linearized by using space-filling curves. Accordingly, the  $B^+$ -tree index is constructed using the linearized values (e.g., the z-values of Z-curve), and the linearized values can be searched by the  $B^+$ -tree search operation.

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

#### **R-trees:**

R-trees are commonly used to handle a range query<sup>1</sup> by finding all objects whose MBRs overlap a range query (Guttman, 1983).

Papadias et al. (1995) propose an R-tree-based framework to handle the topological relations (e.g., *inside* or *covers*) of spatial queries. The framework treats the indexed objects of an R-tree as primary objects and the query object as a reference object. Subsequently, it performs the topological predicates to derive the topological relationships between a reference object and primary objects.

An approach that employs R-trees to handle the directional relations (e.g., *south*) of spatial queries is proposed in (Papadias et al., 1994). First, it applies a pruning strategy to exclude all the MBRs of the R-tree that cannot satisfy the directional relation of a query. Second, the remaining MBRs of the R-tree are tested by using computational geometry methods.

#### **Hashing:**

Belussi et al. propose a hashing approach which allows direct access to the cells of a database grid (Belussi et al., 2002). In particular, a database is divided into cells of a grid, whereupon R\*-trees are applied to cluster the data in each cell. Each of these cells is given a binary and unique hash key to avoid the necessity to search complete paths of the tree structure.

Lo et al. propose Spatial Hash Joins (SHJ) (Lo and Ravishankar, 1996), which consist of two phases: a partition phase and a join phase. In the partition phase, database objects are divided into separate datasets using a spatial partitioning schema. This results in so-called hash buckets, where each object is mapped to its corresponding bucket. Each bucket is identified by its *extent*, i.e., the rectangle of the related partition. In the second phase, buckets which share the same *extent* are joined.

In (Mamoulis and Papadias, 2003), Slot Index Spatial Join (SISJ) is proposed as an extension to SHJs. The key idea is to join indexed spatial data of R-trees with non-indexed spatial data. The SISJ splits the entries of the R-trees into slots and hashes, where each slot has a unique ID. In addition, each slot has identifiers of nodes pointing to their Maximum Bounding Rectangles (MBRs). Non-indexed data is also divided into buckets as they have a similar structure as the slots. In the final step, each bucket is joined with the corresponding nodes in the slot of the R-trees. We note that the

---

<sup>1</sup>Sometimes it is called a window or rectangle query.



proposed approaches apply join operations and computational geometry techniques to answer spatial queries.

In the area of biomedical research, hashing has been applied to index spatial databases containing the biomedical data, such as the topological structure of chemical molecule.

A hash table data-structure has been used to index graphs based on the canonical code (Williams et al., 2007). The canonical code is a string representation of adjacency matrices of a graph (or sub-graph). The canonical code of each sub-graph is then used as the key of the hash table to speed-up the process of isomorphic lookup. However, the authors consider only graph isomorphism with respect to connectivity rather than the labelled variables and edges of subgraphs of spatial databases.

In (Pal and Rao, 2011; Zou et al., 2008) the molecular graph of a spatial database is decomposed into sequences of bit-strings. By means of these bit-strings the connectivity structure of each sequence is captured. In particular, the connectivity structure of each sequence is mapped into true (1) and false (0) values, where 1 denotes that two vertices are connected in a sequence, and 0 denotes that they are not connected. Subsequently, the mapped values are used as a hash key to index all sequences of the decomposed graph. This other similar approaches in biomedical research only consider the connectivity between the nodes of the graph database, but not the location information (e.g., the geographical position (spatial location) of spatial objects).

## 3.2 Clustering

Clustering is a sub-field of data mining and aims at grouping similar (spatial) objects that have similar features into classes. Accordingly, a cluster represents a set of (spatial) objects that are similar to each other, and dissimilar to the objects of other clusters.

Distances are commonly used to measure similarity (or dissimilarity) among the pairs of objects. Minkowski distance is a general measurement on Euclidean space that calculates distance between a set of pairs of points (Han, 2005).

The Minkowski distance is defined in equation 3.1, where  $r > 0$  is the order of the Minkowski metric and  $i = (o_{i_1}, o_{i_2}, \dots, o_{i_n})$  and  $j = (o_{j_1}, o_{j_2}, \dots, o_{j_n})$  are two  $n$ -dimensional data objects.

$$d_r(i, j) = \sqrt[r]{(|o_{i_1} - o_{j_1}|^r + |o_{i_2} - o_{j_2}|^r + \dots + |o_{i_n} - o_{j_n}|^r)} \quad (3.1)$$

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

Note that the Manhattan distance is a special case of Minkowski distance, where  $r=1$ :

$$d_1(i, j) = |o_{i_1} - o_{j_1}| + |o_{i_2} - o_{j_2}| + \cdots + |o_{i_n} - o_{j_n}| \quad (3.2)$$

Similarly, the Euclidean distance is a special case of Minkowski distance, where  $r=2$ :

$$d_2(i, j) = \sqrt{(|o_{i_1} - o_{j_1}|^2 + |o_{i_2} - o_{j_2}|^2 + \cdots + |o_{i_n} - o_{j_n}|^2)} \quad (3.3)$$

The Euclidean distance has the following properties (Han, 2005):  $d_2(i, j) \geq 0$ ,  $d_2(i, i) = 0$ ,  $d_2(i, j) = d_2(j, i)$ ,  $d_2(i, j) \leq d_2(i, k) + d_2(k, j)$ .

According to (Han, 2005; Manolopoulos et al., 2003) clustering methods fall into eight categories: (1) density-based, (2) grid-based, (3) partitioning, (4) hierarchical, (5) model-based, (6) high dimensional, (7) constraint-based, and (8) hybrid (e.g., hierarchical and partition).

Table 3.3 shows a comparison of some of the clustering methods in different categories. As shown in Table 3.3, only the algorithms in the density-based category guarantee a global optimum in terms of clustered data with respect to Euclidean distance. It also indicates that the algorithms in the grid-based category have the fastest processing speeds. Finally, hybrid hierarchical and partition, as well as grid-based approaches have already been applied for clustering qualitative data (Fogliaroni et al., 2011). Hence, this section will focus on the category 1, 2, and 8.

#### 3.2.1 Grid-Based Clustering

In general, a basic grid-based clustering approach divides 2D space into a finite number of rectangular units (Warnekar and Krishna, 1979), where each unit contains at least one object (or point). This approach has the advantage of providing a fast processing, since the speed of clustering process is independent of the number of objects and only depends on the number of units. However, this approach suffers from two considerable weaknesses in terms of the quality of clustered data (Han, 2005): (1) the boundaries of clusters can be either horizontal or vertical and (2) there is no way to detect the diagonal boundary that affects the shape of the cluster, and thus, parts of clusters could be missing and considered in other units.

**Table 3.3:** Comparison between eight categories of clustering methods.

Id	Type	Cluster algorithm	Complexity [time]	Global optima	Requires # of clusters
1	Density-based	DBSCAN (Ester et al., 1996)	$O(n \log(n))$	yes	no
1	Density-based	OPTICS (Ankerst et al., 1999)	$O(n \log(n))$	yes	no
2	Grid-based	Basic approach (Warnekar and Krishna, 1979)	$O(d)$ , $d$ is the # of grids	no	yes
2	Grid-based	WaveCluster (Sheikholeslami et al., 2000)	$O(n)$	no	no
3	Partitioning	K-means (Babu and Murty, 1993)	$O(nkt)$ , $n$ is the # of objects, $k$ is the # of clusters, and $t$ is the # of iterations	no	yes
4	Hierarchical	BIRCH (Zhang et al., 1996)	$O(n)$	no	no
5	Model-based	Expectation-Maximization (EM) (McLachlan and Krishnan, 1997)	$O(n)$	no	no
6	High-Dimensional	CLIQUE (Agrawal et al., 1998)	$O(n)$	no	no
7	Constraint-based	CLTree (Liu et al., 2000)	$O(r)$ , $r$ is the # of regions	no	no
8	Hierarchical and Partition	R-tree (Guttman, 1983)	$O(2^n)$ , $n$ is the # of regions	no	yes

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

However, other grid-based clustering methods such as a Wavelet-based Clustering (WaveCluster) (Sheikholeslami et al., 2000) and a STatistical INformation Grid (STING) (Wang et al., 1997) analyze the distribution of objects and then split the space into units accordingly. Although these methods are considered among the best grid-based clustering methods of their category, they have not been applied to cluster qualitative data so far.

#### 3.2.2 Density-Based Clustering

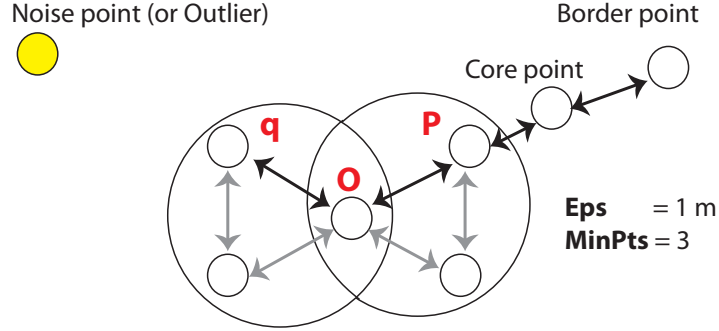
Density-based clustering methods have been developed to detect the arbitrary shapes of dense points or regions. Empirical analysis studies indicate that density-based clustering methods are very practical methods for clustering dense regions based on metric criteria, i.e., Euclidian distance (Ester et al., 1996). Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996), and Ordering Points to identify the Clustering Structure (OPTICS) (Ankerst et al., 1999) are the most popular methods in this category. In this section, we only consider DBSCAN. DBSCAN is a well known clustering algorithm that has been successfully applied to cluster GIS data. DBSCAN separates the areas of high and low density. It uses two parameters: the cluster radius that the points (or polygons) need to lie within ( $Eps$ ), and the minimum number of points ( $MinPts$ ) within a cluster. For clustering purposes, DBSCAN differentiates five types of points; (1) core (Definition 4), (2) border (Definition 5), (3) noise (Definition 6), (4) density-reachable (Definition 7), and (5) density-connected (Definition 8). Following are the definitions of the five points as seen in (Han, 2005):

**Definition 4** (Core point). *A point within  $Eps$  is called a core point if and only if the number of neighborhood objects of this point is greater than or equal  $MinPts$ .*

**Definition 5** (Border point). *A point is called border point if and only if the number of neighborhood objects of this point less than  $MinPts$ .*

**Definition 6** (Noise point). *A point is called noise point when it does not belong to any cluster.*

Figure 3.4 outlines the core, border, and noise points of DBSCAN.



**Figure 3.4:** An example: DBSCAN.

**Definition 7** (Density-reachable point). *A point  $p$  is called density-reachable if the distance between  $p$  and other point  $q$   $\text{dist}(p, q)$  is less than  $Eps$  or if there is a sequence of points  $(p_1, \dots, p_n)$ ,  $p_0 = q$ ,  $p_n = p$  such that  $p_{i+1}$  can be directly reached from  $p_i$ .*

**Definition 8** (Density-connected point). *A point  $p$  is density-connected to a point  $q$  if  $p$  and  $q$  are density-reachable via intermediate object  $O$  with respect to  $Eps$  and  $MinPts$ .*

Based on the previous definitions of density of points, DBSCAN attempts to find the maximum number of density-connected points. Figure 3.4 shows that points  $p$  and  $q$  are density-connected, and thus they can be included in the same cluster<sup>1</sup>.

In the scope of this dissertation, we will use DBSCAN to cluster qualitative data, as it guarantees the global optimum via the reachability concept, and there is no need to specify the number of clusters in advance.

### 3.2.3 Approaches for Clustering Qualitative Data

A lot of research has been undertaken to cluster quantitative data but very little work deals with qualitative data.

R-trees and their modifications have been applied to cluster qualitative data. In (Fogliaroni, 2012; Manolopoulos et al., 2003; Papadias and Manolopoulos, 1997) the Qualitative Spatial Relations (QSRs) between pairs of objects within each MBR have been computed. However, R-trees have two major weaknesses regarding clustering qualitative data:

<sup>1</sup>We note that DBSCAN merges these two clusters into a single cluster after detecting the density-reachable point.

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

1. MBRs can be duplicated, which requires the duplication of QSRs between pairs of nodes (leaves) in several MBRs.
2. By using a number of objects that need to be clustered in advance, R-trees can guarantee only a local optimum with respect to the (Euclidean) distance between objects (Leutenegger et al., 1997). R\*-trees aims to split the number of nodes in a better way, rather than improve the quality of clustered objects.

Fogliaroni et al. (2011) apply a basic grid-based clustering approach to split 2D space into a finite number of rectangular units, where each unit contains an average number of objects (or points). Then QSRs among pairs of objects are abstracted within each unit. Additionally, QSRs are abstracted between the units themselves. Averaging units by the number of objects may improve the quality of produced units (or clusters), but it is not guaranteed. In addition, this approach takes more time than a basic grid-based and inherits the same weaknesses from it, such as repeating the same objects in different grids.

### 3.3 Integrating Qualitative Spatial Reasoning with GISs

Egenhofer and Mark (1995) propose a research agenda called Naive Geography which uses formal theories and models of human common-sense reasoning about geographic space and time. In particular, the authors argue that Naive Geography should employ qualitative spatial reasoning as the basis of intelligent Geographic Information Systems (GISs) in order to be suitable for non-expert GIS users to use GISs without any specialized training. Following the idea of Naive Geography, several approaches have been proposed in recent years for integrating Qualitative Spatial representation and Reasoning (QSR) with GISs. The approaches can be divided into two categories: intuitive interaction with GISs (see Section 3.3.1) and matching geo-spatial information (see Section 3.3.2).

#### 3.3.1 Approaches for Intuitive Interactions

In (Egenhofer and Franzosa, 1995), the 9-Intersection Model (9IM) was integrated with GISs, so that topological relations such as *contains* and *intersect* could be computed. Later, the 9IM was extended as a topological and spatial extension of SQL (Egenhofer, 1994b) to query Spatial Data-Bases (SDBs) of GISs.

### 3.3 Integrating Qualitative Spatial Reasoning with GISs

---

Egenhofer (1997) proposes a Spatial-Query-By-Sketch (SQBS) approach for querying SDBs using a sketch-based interface. In this approach, users build spatial queries by sketching configurations that they are in search of. The sketched-query is then transformed into a set of scene networks, where each pair of objects is connected via spatial relation(s). Additionally, cardinal directions (Frank, 1992) and the 9IM (Egenhofer and Franzosa, 1995) are used to represent qualitative spatial relations. The SQBS approach concentrates on translating a sketched-query into qualitative information rather than matching it against SDBs.

Several visual tools are developed on the basis of the SQBS approach. For example, visual tools for querying the SDBs by means of a sketch-based interface are presented in (Blaser and Egenhofer, 2000; Kopczynski, 2006). Users first need to sketch their queries by drawing the objects, and then these queries are used to retrieve relative results from SDBs. A similar visual approach is taken in (Caduff and Egenhofer, 2005), where they take into account varying capabilities of mobile devices. However, the main focus is on Human-Computer-Interaction (HCI) level rather than on the matching of corresponding objects of SDBs.

#### 3.3.2 Approaches for Matching Geo-Spatial Information

In (Wallgrün et al., 2010) the authors propose a matching approach based on QSR. They assume a database  $\mathcal{D}$  given as a qualitative constraint network ( $\mathcal{G}_{\mathcal{D}}$ ), i.e. a directed graph with relational labels. Given a sketched user query, a qualitative constraint network  $\mathcal{G}_{\mathcal{Q}}$  is derived.  $\mathcal{G}_{\mathcal{Q}}$  is then matched against  $\mathcal{G}_{\mathcal{D}}$  by finding all constraints in  $\mathcal{G}_{\mathcal{D}}$  that satisfy possible binary object tuples of  $\mathcal{G}_{\mathcal{Q}}$ . The authors only consider qualitative direction relations among street junctions as points in a small toy domain, while we consider extended objects.

Similarly, Chipofya (2011) presents a method that finds optimal matches among multiple sketch maps, where qualitative spatial relations among objects of each sketch map are represented as a labelled graph. Then the graphs of all sketch maps are matched and the optimal matches are detected. In the works (Chipofya, 2011; Wallgrün et al., 2010), the authors concentrate on finding solutions of the graphs at run time.

In (Bruns and Egenhofer, 2000) the authors state query matching as a Constraint Satisfaction Problem (CSP) based on so called scenes, which are defined by a set of spatial relations between objects. A sketched user query is considered as a scene that

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

can be matched against subsets (or scenes) of a database. Matching is then the pairing of a query with the constraints and variables of subsets of all of the maximal complete subgraphs (maximal cliques) of the database. However, the authors did not consider any technique for accelerating the search in SDBs.

In (De Felice et al., 2011), a hybrid qualitative-quantitative spatial reasoning and matching approach is proposed. It integrates qualitative information with quantitative information stored in SDBs. Based on incomplete qualified information in SDBs and quantitative information, the system applies QSR and matching techniques to integrate new qualitative information with the quantitative information in SDBs.

#### 3.4 Summary

To conclude this chapter, Spatial Data-Base Management Systems (SDBMSs) of Geographic Information Systems (GISs) play a crucial role in answering spatial queries efficiently.

SDBMSs incorporate efficient spatial data models to store and manipulate spatial data. Currently, the Geometry Object Model (GOM) is the predominant spatial data model in GISs. However, the GOM and other similar models do not support different types of spatial relations other than the topological ones.

SDBMSs apply R-trees, B-trees, and hashing indexing methods to speed-up answering spatial queries. B<sup>+</sup>-trees are used with the space-filling curves to process spatial queries. However, the linearization process of the space-filling curves requires an exponential number of steps to enumerate all objects in 2D space (Faloutsos and Roseman, 1989). In turn, R-trees and their variants suffer from two major weaknesses: (1) the spatial relations among the database objects need to be computed at run time, which is computationally expensive and (2) join operations are required to process spatial queries that are expensive in terms of time. Hashing methods are combined with R-trees to answer spatial queries quickly. Therefore, they inherit the same weaknesses as R-trees.

Aside from the mentioned weaknesses, the indexing methods are designed to only handle a single aspect of space such as topology or direction.

Clustering methods can be applied to spatial databases to cluster qualitative data. Although many clustering methods exist, only the grid-based and hierarchical methods have been applied to cluster qualitative data. The applied methods generate low quality



clusters, since they do not guarantee a global optimum. Additionally, the same objects can be included in different clusters.

Qualitative Spatial representation and Reasoning (QSR) methods are combined with GISs to facilitate an intuitive and qualitative interaction between GISs and their users. QSR methods use matching approaches to speed-up answering qualitative spatial queries in the graph databases of GISs. The main drawback of these approaches is that the matching process is done at run time, which is computationally expensive. The matching requires an exponential number of steps to enumerate all possible matches to queries.

### 3. SPATIAL INFORMATION MANAGEMENT AND SYSTEMS

---

## Querying, Reducing, and Matching Qualitative Information

Geographic Information Systems (GISs) do not adequately allow users to query spatial databases by means of qualitative descriptions such as *left*, *north of*, or *inside*. Such qualitative descriptions can be used to formulate Qualitative Spatial Queries (QSQs) (see Section 4.1). In order to enable qualitative spatial query processing, we integrate qualitative spatial models into GISs (see Section 4.2). We abstract binary Qualitative Spatial Relations (QSRs) from database objects and store them in a Qualitative Spatial Layer (QSL) to avoid computing QSRs for every single query (see Section 4.3). Next, we consider the spatial query answering problem as a sub-graph isomorphism matching problem (see Section 4.3.2). As abstracting the QSL results in a high space complexity in terms of qualitative representations, we apply two qualitative data reduction strategies (see Section 4.4): (1) reduction by clustering and (2) reduction by a converse operation.

### 4.1 Qualitative Spatial Queries

Qualitative Spatial Queries (QSQs) are *non-geo-referenced* queries that aim to query spatial databases qualitatively and intuitively. Although QSQs do not possess geographic locations, they aim at providing services for finding locations (e.g., restaurants or damaged buildings). Additionally, the QSQs are limited to querying categories/classes of objects (e.g., rivers) rather than individuals (e.g., the “Weser” river), since these categories/classes of objects are commonly used by people.

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

Moreover, we only consider objects of atomic categories such as river or building, and no higher order ontological categories such as state or country, which may summarize several atomic objects<sup>1</sup>. In particular, the QSQs usually come in the form of constraints between objects. The constraints are usually binary Qualitative Spatial Relations (QSRs) holding between pairs of objects. In the simplest case, the QSQs contain a single qualitative spatial relation and come in the form:

$$\{\text{a reference object, a qualitative spatial relation, a primary object}\}$$

For example, in a spatial query such as “Find a *restaurant inside a park*”, *inside* is the qualitative spatial relation that holds between the primary object *restaurant* and the reference object *park*. In order to answer QSQs, all of the constraints of a spatial database that satisfy the constraints of the QSQs need to be enumerated, since the geographic locations are not given or known. For instance, answering the spatial query mentioned in the previous example results in testing all the object pairs restaurant-park stored in a spatial database.

The QSQs fall into two categories (see Figure 4.1(a)): (1) verbal descriptions<sup>2</sup> and (2) visual descriptions<sup>3</sup>. The main distinction between these categories is that spatial relations are explicitly expressed in verbal descriptions, whereas spatial relations are implicit in visual descriptions, and hence need to be abstracted. Consider two examples in Figure 4.1(a), where the spatial relation *inside* appears explicitly in the verbal description, but implicitly in the visual descriptions.

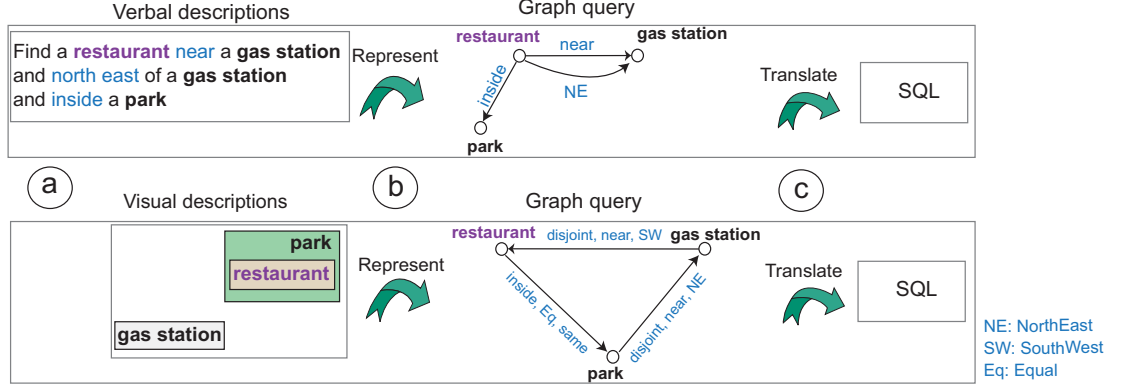
The QSQs may contain two or more binary QSRs. Hence, a qualitative spatial query is considered as a labelled graph  $\mathcal{G}_Q = \langle V_Q, E_Q \rangle$ , where  $V_Q$  denotes the objects (vertices) of a graph and  $E_Q$  denotes labelled edges (constraints) holding among these objects. In Figure 4.1(b), we depict two possible graph query representations for verbal and visual descriptions of queries respectively. It is worth mentioning that visual descriptions of a QSQ could be represented differently. For instance, the topological relation holding a restaurant and a park can be considered as *contains* instead of *inside*, i.e., a park

---

<sup>1</sup>Dealing with the semantics or ontologies of objects (e.g., a hospital is a *part of* district) is beyond the scope of this dissertation.

<sup>2</sup>Dealing with the natural language processing issues of the verbal descriptions of the QSQs is beyond the scope of this dissertation.

<sup>3</sup>In the literature, these kind of queries are commonly called proctorial queries as well.



**Figure 4.1:** An example: qualitative spatial query formalism.

*contains* a restaurant. We note that representing the QSQs as graph queries is crucial and plays a fundamental role in explicitly identifying the components of queries (e.g., object pairs). This representation allows for easily translating queries into Structured Query Language (SQL) and for identifying appropriate matching procedures to efficiently answer such queries, e.g., the spatial query answering problem can be considered as a sub-graph isomorphism matching problem.

Finally,  $\mathcal{G}_Q$  needs to be translated into SQL and processed on a Spatial Data-Base Management System (SDBMS) (see Figure 4.1(c)). The translation operation is done automatically by detecting each pair of  $\mathcal{G}_Q$ . Then the reference object, the primary object, and the relation of the qualitative model of each pair are assigned to appropriate fields in SQL, which are related to fields in database tables.

An example of an SQL translation of verbal descriptions of a QSQ is illustrated in Figure 4.2.

## 4.2 Enabling Qualitative Spatial Queries in GISs

In section 3.1.1.1, we pointed out that several spatial data models have been developed to model and manage spatial data in SDBMSs. In addition, the Geometry Object Model (GOM) has been recognized in (Manolopoulos et al., 2005) as one of the most practical spatial models, as it provides a rich set of functionalities to store and manipulate spatial data. However, like other spatial data models, GOM only integrates a topological model: the Dimensionally Extended 9-Intersection Model (DE-9IM), which offers eight topological predicates (e.g., *Disjoint*) to compute the topological relations between

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

```

SELECT T,F

FROM ( SELECT centroid, id
      FROM bremenosm
      WHERE RO='restaurant' AND PO='gas station'
      AND ComputeDistance(RO.geometry, PO.geometry)='near'
      AND ComputeDirection(RO.geometry, PO.geometry)='NorthEast'

    ) AS T
,
FROM ( SELECT centroids, id
      FROM bremenosm
      WHERE RO='restaurant' AND PO='park'
      AND ComputeTopology(RO.geometry, PO.geometry)='inside'

    ) AS F

WHERE T.RO=F.RO

```

RO: Reference Object  
 PO: Primary Object

**Figure 4.2:** An example: verbal descriptions of a QSQ are translated to SQL.

pairs of geometric objects. However, QSQs may contain different kinds of QSRs (e.g., direction) other than topology. Hence, GOM cannot cope with QSQs that contain QSRs other than topology such as direction or distance. Recall the SQL example in Figure 4.2 that shows that the directional and distance predicates are essential to process the SQL. Following (Freeman, 1975) and (Egenhofer, 1997), spatial relations which are beneficial for developing several real life applications including Geographic Information Systems (GISs) fall into three categories: (1) topological, (2) directional, and (3) distance. In order to enable qualitative formalism (e.g., QSQs) of direction and distance relations, we integrate two kinds of qualitative spatial models into GOM: (1) the Cardinal Direction Models (CDMs) and (2) the Absolute Distance Model.

##### (1) The Cardinal Direction Models

In the context of cognitive science, cognitive studies by (Franklin and Tversky, 1990; Franklin et al., 1995; Huttenlocher et al., 1991; Wang and Schwering, 2009) give strong evidence to support the cognitive plausibility of the cone-based model. In addition, the

cone-based model is a practical approach (Skiadopoulos et al., 2007), due to its ability to retrieve the directional relations based on the verbal descriptions of users. For these reasons, the cone-based model has been used in GISs (Clementini and Billen, 2006). Therefore, we integrate the cone-based model into GOM to serve the QSQs that are given by means of verbal descriptions.

In the context of GIS, the Cardinal Direction Model (CDM) for extended objects is strongly suggested by (Blaser and Egenhofer, 2000; Bruns and Egenhofer, 2000; Egenhofer, 1997; Skiadopoulos and Koubarakis, 2004). In particular, the authors conclude that the CDM should be used to answer visual-based queries (e.g., Spatial-Query-by-Sketch (Egenhofer, 1997)) due to its ability to capture meaningful distinctions of space (e.g., Sketched-queries inherently involve geometric information). This implies more relevant results to the queries. We thusly integrate the CDM into GOM to serve the QSQs that are given by means of visual descriptions.

### (2) The Absolute Distance Model

In Section 2.2.3, we indicated that the notion of distance is *context-dependent* and influenced by several factors, e.g., “attractiveness”. Therefore, an ideal distance model would take these factors into account. However, in this dissertation, for simplicity, we only consider the Euclidean distance to develop *context-independent* distance model.

Based on the qualitative distance model (Gahegan, 1995), we propose an absolute distance model that assigns one of the four distance relations: *ZeroDist*, *near*, *medium*, and *far* to pairs of objects in  $\mathcal{D}^2$ . This representation is based on the Minimum Bounding Rectangle (MBR).

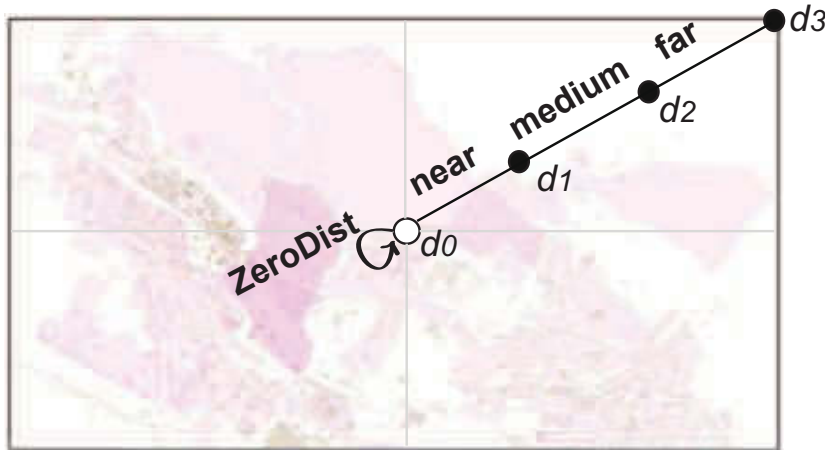
**Definition 9** (the Absolute Distance Model).

Let  $P_{cent} = (x_{cent}, y_{cent})$  denote the centroid of the MBR. In addition,  $d_{max} = d_3$  denotes the maximum distance between  $P_{cent}$  and one of the corners of the MBR. In order to define the four relations, we need two additional distance values  $d_1$  and  $d_2$  with  $d_0 < d_1 < d_2 < d_3$ .  $d$  denotes the distance between either the point of interest and  $P_{cent}$  or between any two points of interest. The relation  $Dist_r$  is considered:

$$Dist_r = \begin{cases} ZeroDist & \text{if } d = 0; \\ near & \text{if } d_0 < d \leq d_1; \\ medium & \text{if } d_1 < d \leq d_2; \\ far & \text{if } d_2 < d < d_3. \end{cases}$$

## 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---



**Figure 4.3:** The distance model with four relations.

In Figure 4.3 we depict a distance model with a fixed region of interest and an equidistant partition scheme using all  $d_i$ . We integrate the absolute distance model into GOM to allow for deriving of distance relations between pairs of geometric objects.

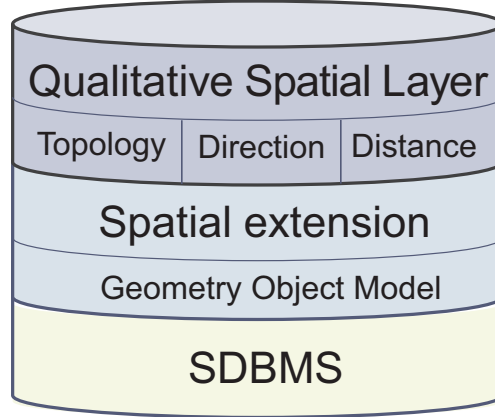
It is crucial to mention that GOM supports R-trees indexing (for more details about R-trees, see Section 3.1.2.2). In this dissertation, we use R-trees to accelerate computing the topological relations among geometric objects.

### 4.3 Extending a Qualitative Spatial Layer into GISs

Abstracting Qualitative Spatial Relations (QSRs) from geometries becomes an impractical approach when the abstraction process needs to be conducted for every single query. In this dissertation, we call this kind of approach a **naive approach**, in which all QSRs holding among database objects need to be abstracted and matched to the QSRs of QSQs. Another possibility includes a buffered approach, in which previous query matches are memorized in order to avoid recomputing QSRs. However, these approaches require a massive amounts of database storage, as the number of matches of processed queries is usually very large. Moreover, many QSQs may not be processed previously which implies that the **naive approach** needs to be applied to answer these queries. In other words, these approaches do not guarantee the acceleration of processing QSQs.

Alternatively, QSRs can be abstracted and explicitly stored in a Qualitative Spatial Layer (QSL) of spatial databases to avoid the additional cost of the abstraction process





**Figure 4.4:** A logical view of the qualitative database layer extension.

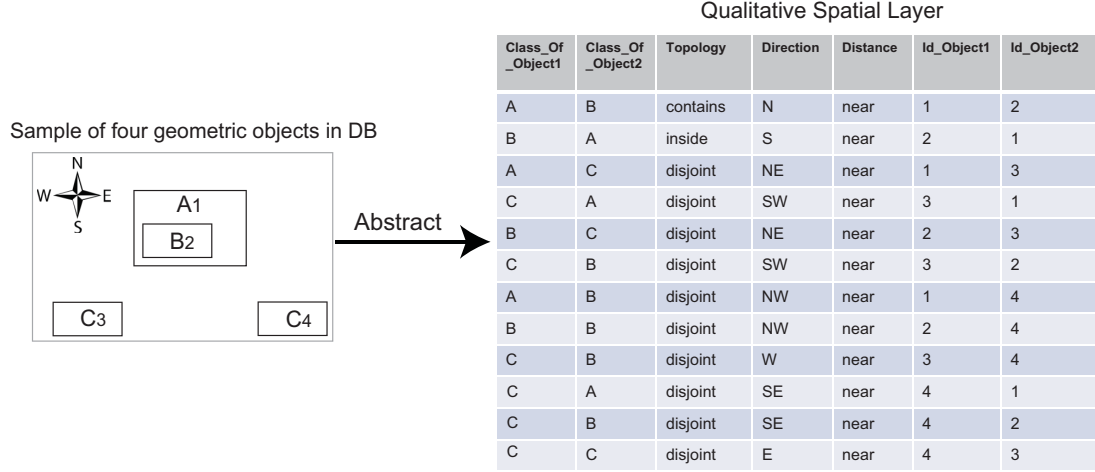
every time (Fogliaroni et al., 2011). Therefore, as shown in Figure 4.4, we abstract the QSL and store it once as an extra database storage layer upon the spatial layer of a SDBMS. In the QSL extension, the structure, the capabilities, and the functionalities of GOM and the relational SDBMS are fully preserved, but directional and distance predicates are extended to the GOM and SQL. The QSL is based on an abstraction process that computes the qualitative relations between all pairs of objects.

Figure 4.5 shows the database table of a qualitative spatial layer abstracted from four geometric objects  $\{A_1, B_2, C_3, C_4\}$ .

Consider a database  $\mathcal{D} = \langle O_{\mathcal{D}}, F_{\mathcal{D}} \rangle$ , where  $O_{\mathcal{D}}$  denotes the set of objects occurring in the database, and  $F_{\mathcal{D}}$  indicates their connected geometric features (e.g., their centroid positions, geometric type, etc). Then, pairs of objects  $(o_i, o_j) \in O_{\mathcal{D}}$  and their relational structure can be represented as a complete directed labelled graph  $\mathcal{G}_{\mathcal{D}} = \langle V_{\mathcal{D}}, E_{\mathcal{D}} \rangle$ , where  $E_{\mathcal{D}}$  denotes labelled edges (qualitative spatial relations) that hold among objects or vertices ( $V_{\mathcal{D}}$ ). The order of a  $O_{\mathcal{D}}$  is denoted by  $|O_{\mathcal{D}}|$ , which is the number of objects. For instance, assume that there are two objects ( $X$  and  $Y$ ) and a binary topological relation *disjoint* holding among them; the direction of the edge must be considered as  $X\_Disjoint\_Y$ , which has a different label than  $Y\_Disjoint\_X$ . Algorithm 1 generates a complete graph  $\mathcal{G}_{\mathcal{D}}$  from  $O_{\mathcal{D}}$  in  $O(\eta|O_{\mathcal{D}}|^2)$  steps, where  $\eta$  is a number of qualitative models (or calculi). The algorithm has a complexity of  $O(\eta|O_{\mathcal{D}}|^2)$  space as well. It iterates over all pairwise *disjoint* object pairs  $O_{\mathcal{D}}$  (lines 2 to 4), then calculates the relation<sup>1</sup> (line 5), and adds the edges and their labels to  $\mathcal{G}_{\mathcal{D}}$  (line 6).

<sup>1</sup>In the scope of this dissertation, standard qualitative spatial models are applied (c.f Section 2.2).

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION



**Figure 4.5:** Example: a qualitative spatial layer that represents all the binary qualitative spatial relations (per each one of the three qualitative models including topology, direction, and distance) among four geometric objects.

---

**Algorithm 1:** AbstractDataBaseGraph(*Objects*  $O_{\mathcal{D}}$ , *ObjGeometries*  $F_{\mathcal{D}}$ )

---

**input** :  $O_{\mathcal{D}}$ : all database objects

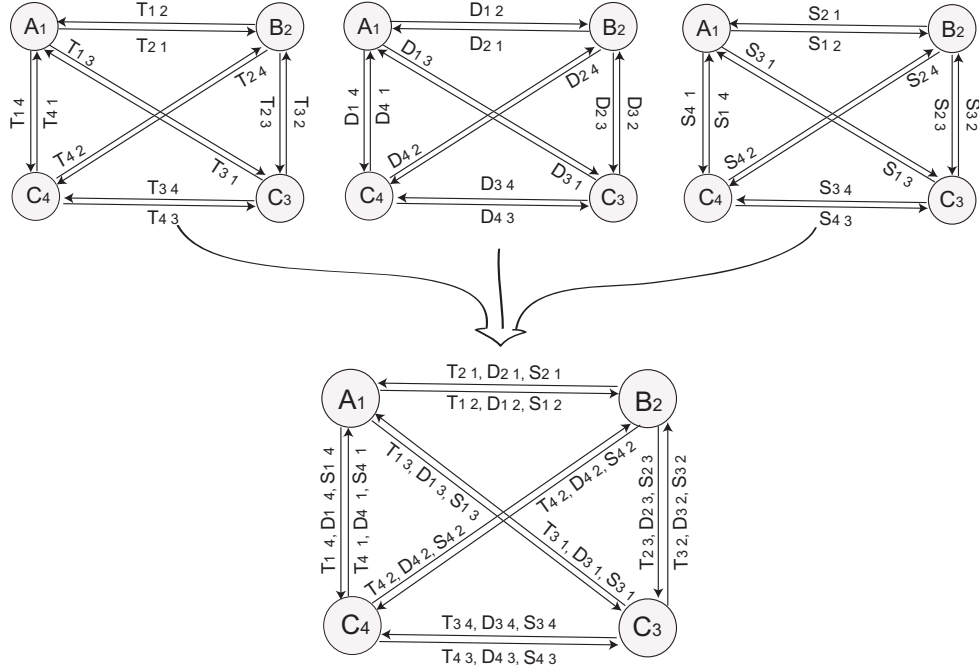
**output**:  $\mathcal{G}_{\mathcal{D}}$ : a complete graph database that contains all pairs of objects and their corresponding relations

```

1 initialization:  $r \leftarrow \text{NULL}$ ;  $\mathcal{G}_{\mathcal{D}} \leftarrow \text{NULL}$ ;
2 for  $i \leftarrow 1$  to  $|O_{\mathcal{D}}|$  do
3   for  $j \leftarrow 1$  to  $|O_{\mathcal{D}}|$  do
4     if  $(o_i.id \neq o_j.id)$  then
5        $r \leftarrow \text{ComputeQSRels}(f_i.geometry, f_j.geometry)$ ;
6        $\mathcal{G}_{\mathcal{D}}.\text{add} \leftarrow (o_i, r, o_j)$ ;
7     end
8   end
9 end
10 return  $\mathcal{G}_{\mathcal{D}}$ ;

```

---



**Figure 4.6:** The  $QCND$  using three qualitative models T, D, and S.

#### 4.3.1 Multi-Graph Representations

As we mentioned in the previous section, the complete graph database  $\mathcal{G}_D$  is the result of the abstraction process.  $\mathcal{G}_D$  represents several graphs that are combined into a single graph. Each graph represents object pairs associated with a single qualitative relation of a qualitative model. If we consider the abstracted binary relations as only labels, then each edge of  $\mathcal{G}_D$  can hold multi-labels, where each label of an edge belongs to a spatial relation of a qualitative model. For reasoning purposes (e.g., composition), in the scope of this dissertation we make use of a multi-Qualitative Constraint Network ( $QCND$ ) which combines  $m$  Qualitative Constraint Networks (QCNs) (each corresponding to a specific qualitative model) into a single QCN (cf. Section 2.3), where  $m$  is the number of qualitative models considered. As depicted in Figure 4.6,  $QCND$  can also be represented as a labelled directed graph where each edge is labelled by the concatenation of the labels of the single QCNs (Topology (T), Direction (D), and Distance (S)).

---

Dealing with errors due to noisy data and imprecise numerical representations is beyond the scope of this dissertation.

## 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

### 4.3.2 Matching a Qualitative Spatial Layer

Given a database  $\mathcal{D}$  and a qualitative query  $\mathcal{Q}$ , we can interpret both as graphs:  $\mathcal{G}_{\mathcal{D}}$  and  $\mathcal{G}_{\mathcal{Q}}$ . As we indicated in (Al-Salman et al., 2012), the spatial query answering problem can be considered as a sub-graph isomorphism matching problem given the above information. We denote the order of a graph  $\mathcal{G}_X$  by  $|\mathcal{G}_X|$ , which is the number of vertices, i.e.  $|\mathcal{G}_X| = |V_X| = |O_X|$ . Two labels in a graph are considered equal if and only if the intersection of two corresponding constraints is not empty.  $\mathcal{G}_{\mathcal{Q}}$  is considered to be a sub-graph which contains  $n = |\mathcal{G}_{\mathcal{Q}}|$  variables and will be matched against subsets of the complete graph ( $\mathcal{G}_{\mathcal{D}}$ ), where each subset contains exactly the same number of variables as  $\mathcal{G}_{\mathcal{Q}}$ .

**Definition 10** (exact match).

*A user query is said to be exactly matched if the intersection between all of the constraints of  $\mathcal{G}_{\mathcal{Q}}$  and all constraints of a subset of  $\mathcal{G}_{\mathcal{D}}$  is not empty.*

**Definition 11** (partial match).

*A user query is said to be partially matched if the intersection between a constraint of  $\mathcal{G}_{\mathcal{Q}}$  and a constraint of a subset of  $\mathcal{G}_{\mathcal{D}}$  is not empty.*

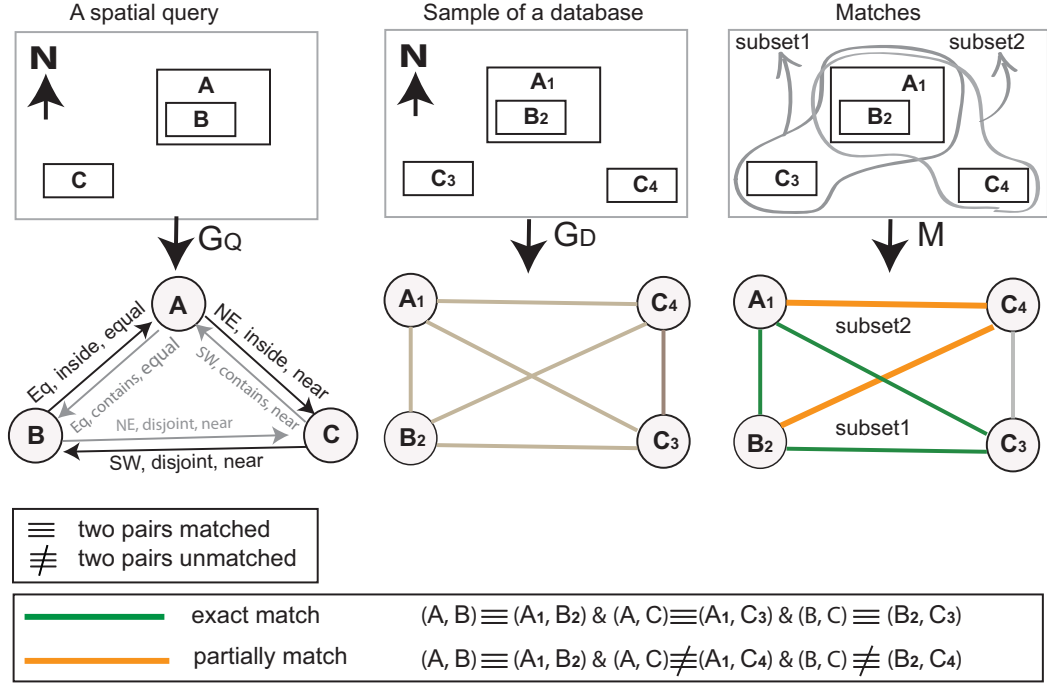
**Definition 12** (no match).

*A user query is said to be not matched if the intersection between all of the constraints of  $\mathcal{G}_{\mathcal{Q}}$  and all constraints of every subset of  $\mathcal{G}_{\mathcal{D}}$  is empty.*

In Figure 4.7 we depict an example of how the variables of  $\mathcal{G}_{\mathcal{Q}}$  can be matched against the variables of  $\mathcal{G}_{\mathcal{D}}$  based on three qualitative models.

In this dissertation, we will only focus on the exact matching between  $\mathcal{G}_{\mathcal{Q}}$  and  $\mathcal{G}_{\mathcal{D}}$ , e.g., we do not consider the conceptually neighboring queries.

Finding sub-graph isomorphisms requires an exponential number of steps to enumerate all the possible exact matches. A common way to deal with sub-graph isomorphisms is to represent the possible matches as an interpretation tree (*Itree*) (Andrew, 1990). The leaves of the *Itree* keep all possible matches between  $\mathcal{G}_{\mathcal{Q}}$  and  $\mathcal{G}_{\mathcal{D}}$  (see Figure 4.8). The *Itree* is a valuable representation since it allows for several search techniques and heuristics to be applied (e.g., breadth first search or A-star) in order to prune the search space of the tree, such as by using a Breadth-First Search (BFS) (Cormen et al., 2009).



**Figure 4.7:** Matching  $\mathcal{G}_Q$  against  $\mathcal{G}_D$ : the first subset is exactly matched by users query, the second subset is partially matched, where the pairs  $\{(A, C), (A_1, C_4)\}, \{(B, C), (B_2, C_4)\}$  differ by a directional relation.

The Qualitative Layer Matcher (QLM) algorithm is depicted in Algorithm 2. In general, QLM matches  $\mathcal{G}_Q$  against the *Itree* by using BFS. Given  $\mathcal{G}_Q$  and  $\mathcal{G}_D$ , it starts from the root of the *Itree* by expanding each object of  $\mathcal{G}_D$  as broadly as possible (lines 2 to 5). In order to determine the admissible sub-branches of the *Itree*, the QLM first enforces a unary constraint to check if the label of object  $q_i$  is matched to  $o_i$ . If a unary constraint is satisfied, it then enforces a binary constraint. Specifically, it checks if the intersection between the constraints of the pairs of objects  $(q_i, q_j) \in \mathcal{G}_Q$  and  $(o_i, o_j) \in \mathcal{G}_D$  is not empty (line 3). Therefore, any sub-branch that does not satisfy the unary and binary constraints can be pruned. Furthermore, if all the constraints of the sub-branches of the *Itree* do not match with the corresponding sub-branches of  $\mathcal{G}_Q$ , then the whole search process is terminated, since no solution can be found in  $\mathcal{G}_Q$  (lines 6 and 7). Otherwise, this process keeps expanding the *Itree* branches until the number of *Itree* levels reaches the number of objects of  $\mathcal{G}_Q$  (lines 8 to 11). Simplified and clarified,

## 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

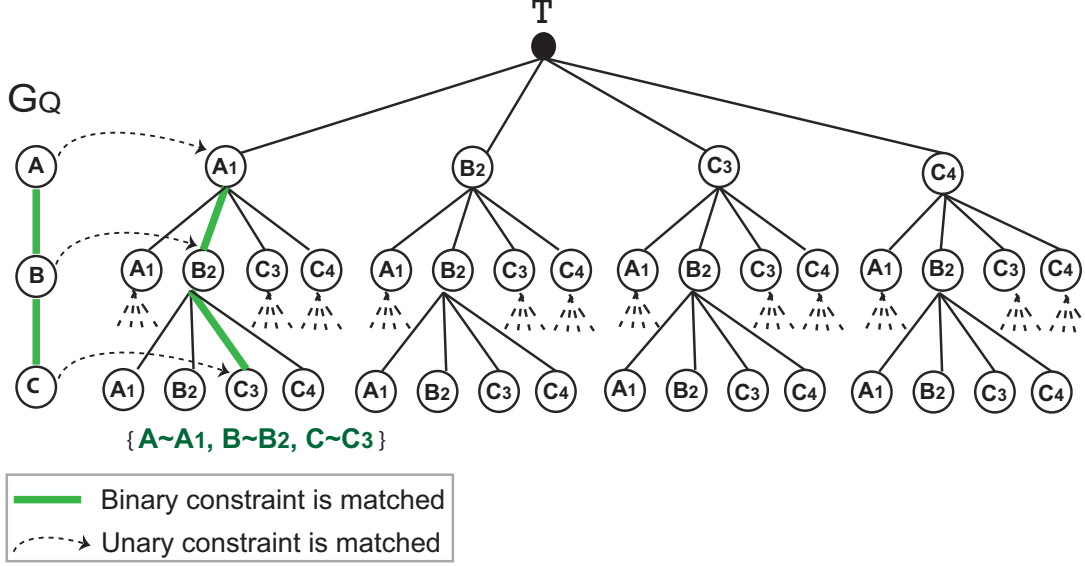


Figure 4.8: Matching the *Itree* to the unary and binary constraints of  $G_Q$ .

Figure 4.8 shows an example of matching both unary and binary constraints of the *Itree* by a  $G_Q$ .

### 4.4 Qualitative Data Reduction

The abstraction process of  $G_D$  results in a high space complexity in terms of the amounts of qualitative data. In order to cope with this complexity, we apply two qualitative data reduction strategies: (1) Qualitative Data Reduction by Clustering (Section 4.4.1) and (2) Qualitative Data Reduction by a Converse Operation (Section 4.4.2).

#### 4.4.1 Qualitative Data Reduction by Clustering

Generally speaking, every object is spatially related to every other object in 2D space. Following the first law of geography that states “everything is related to everything else but nearby things are more related than distant things” (Tobler, 1970), it makes sense to group nearby objects into clusters.

Here, the main purpose of clustering is to reduce the amounts of qualitative data. To do this, database objects are spatially represented by clusters and then the QSRs are abstracted within the corresponding clusters and between clusters themselves.

---

**Algorithm 2:** Qualitative\_Layer\_Matcher(*DBgraph*  $\mathcal{G}_D$ , *Query*  $\mathcal{G}_Q$ )

---

**input** :  $\mathcal{G}_D$ : a complete graph database,  $\mathcal{G}_Q$ : user's query  
**output**:  $\mathcal{M}$ : a set of matches satisfying  $\mathcal{G}_Q$   
1 initialization:  $\Upsilon \leftarrow \text{NULL}$ ;  $Itree \leftarrow \text{NULL}$ ;  $\mathcal{M} \leftarrow \text{NULL}$ ;  $\ell \leftarrow |\mathcal{G}_Q|$ ;  
2  $Itree.\text{Level}(1) \leftarrow \mathcal{G}_D$ ;  
3  $\Upsilon \leftarrow Itree.\text{Level}(1).\text{GetMatch}(\mathcal{G}_Q.\text{Level}(1))$ ;  
4  $\mathcal{M} \leftarrow Itree.\text{Level}(1).\text{GetMatch}(\mathcal{G}_Q.\text{Level}(1))$ ;  
5 **for**  $i \leftarrow 2$  **to**  $\ell$  **do**  
6     **if**  $\Upsilon == \text{NULL}$  **then**  
7         EXIT;  
8     **else**  
9          $\Upsilon \leftarrow Itree.\text{Level}(i).\text{GetMatch}(\mathcal{G}_Q.\text{Level}(i))$ ;  
10          $\mathcal{M} \leftarrow \mathcal{M}_{i-1} \uplus \Upsilon$ ;  
11     **end**  
12 **end**

---

Consequently, the total space consumption might be reduced with respect to the original size of the graph database ( $\mathcal{G}_D$ ).

In his dissertation, Fogliaroni (2012) applies grid-based clustering algorithms, in which qualitative spatial relations are computed among objects inside grids and among clusters themselves to reduce the amount of qualitative data. This process guarantees the retrieval of the same results as without clustering. The author further points out that some binary relations can be inferred from (each) other based on converse and composition properties of qualitative spatial calculi. Accordingly, inferable relations do not need to be stored in a database and are created at run time, which implies reduction of stored qualitative data. For example, if we know that two clusters are *disjoint* from each other, then we can directly conclude that all objects in the clusters are *disjoint* from each other as well. This implies that only *disjoint* relations between clusters need to be computed and stored, instead of all of the *disjoint* relations between individual objects in the clusters.

We apply clustering for the same purpose, albeit with a focus on three specific aspects: (1) the density of clusters, (2) enclosing clusters (cf. Section 4.4.1.1), and (3) computing the QSRs between the clusters in a more sophisticated manner (see Section 4.4.1.3). As we indicated in Section 3.2, a grid-based approach suffers from two weaknesses:

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

(1) the same object could be included in several grids or clusters and (2) it does not consider the spatial density of objects in 2D space. Furthermore, we pointed out that the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996) includes two advantageous properties that others do not. Firstly, it does not require one to specify the number of clusters in advance. Secondly, it guarantees a global optimum in terms of clustered data based on the concept of reachability (cf. Section 3.2.2). Therefore, we apply DBSCAN on the database objects  $O_{\mathcal{D}}$ .

Given a set of  $n$  points of geometric objects  $C = \{c_1, c_2, c_3, \dots, c_n\}$  stored in  $\mathcal{D}$ ,  $C$  is partitioned into  $z$  number of *disjoint* clusters  $C_i$ 's contained by  $F$  and satisfying the following conditions:

- $F = \bigcup_{i=1}^z C_i$ ,  $C_i \cap C_j = \emptyset$ , (for  $i \leq 1$ ,  $j \leq z$ ,  $i \neq j$ ),
- each cluster  $C_i$  represents points with a label  $i$ .

We note that we consider points instead of regions in clustering process to accelerate the process. In the next section, we will describe the strategies of enclosing clusters which will guarantee that all region extents will be included in the clusters.

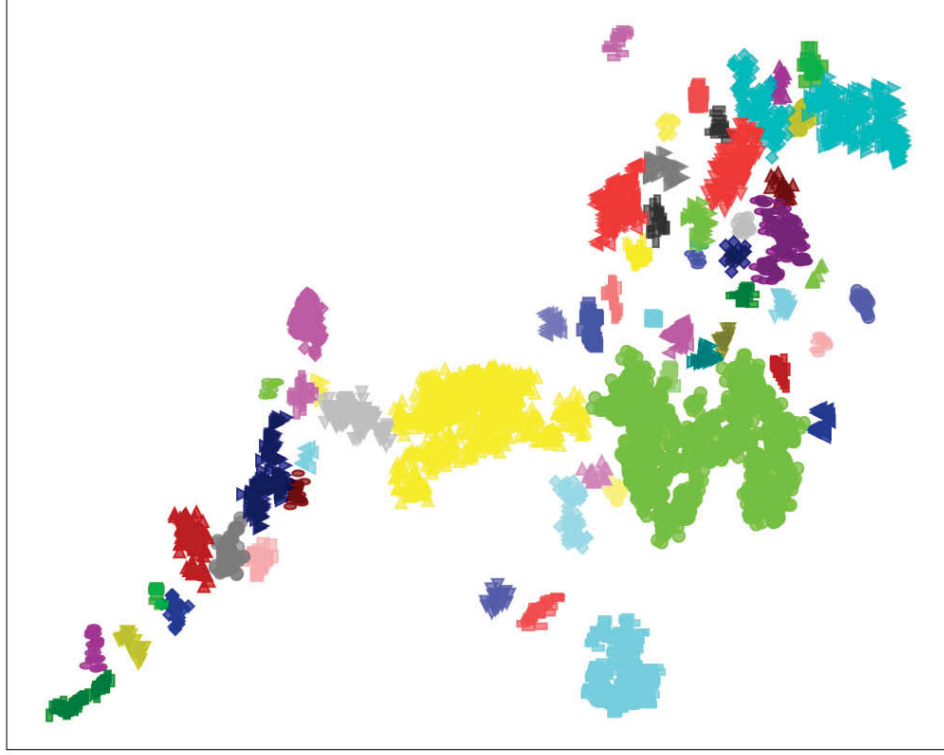
In the rest of this chapter, we will make use of  $|C|$  to denote the order of clusters  $C$ , which indicates the number of clusters. Additionally, the order of a cluster  $C_i$  will be denoted by  $|C_i|$ , which indicates the number of objects occurring in a cluster.

In general, DBSCAN separates the areas of high density from low density. It uses two parameters: the radius of a cluster where the points (or polygons) need to be included in (*Eps*) and the minimum number of points (*MinPts*) within a cluster. Figure 4.9 shows objects in Bremen that are clustered using DBSCAN, where the overlaid plots (polygons) of *the same color and shape* belong to the same cluster. Based on DBSCAN, the qualitative data reduction can be achieved. The equation of qualitative data reduction rate of  $\mathcal{G}_{\mathcal{D}}$  by DBSCAN is given (in percent) by:

$$rate(\mathcal{G}_{\mathcal{D}}) = \left[ 1 - \frac{(\sum_{i=0}^z R_w^i + R_b)}{|\mathcal{G}_{\mathcal{D}}|} \right] \cdot 100 = \left[ 1 - \frac{(\sum_{i=0}^z (|C_i|^2 - |C_i|)) + (|C|^2 - |C|)}{|\mathcal{G}_{\mathcal{D}}|} \right] \cdot 100 \quad (4.1)$$

where  $z$  is the number clusters,  $|\mathcal{G}_{\mathcal{D}}|$  is the number of spatial relations between all object pairs (without considering clustering),  $R_w^i$  is the number of spatial relations between *object pairs* within an individual cluster, and  $R_b$  is the number of spatial





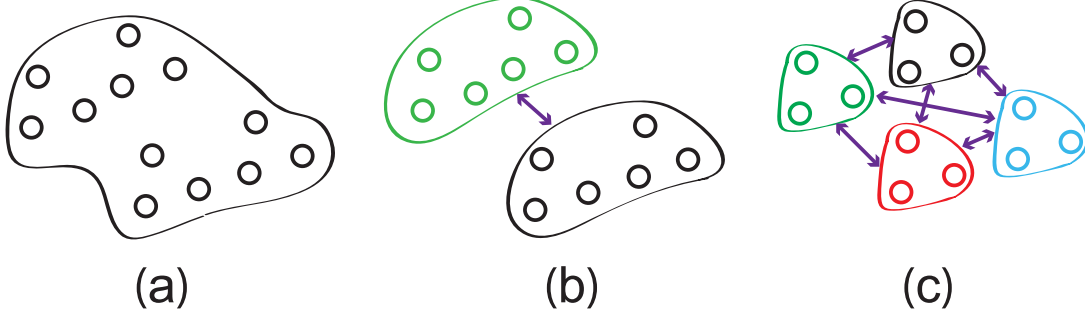
**Figure 4.9:** A clustering of the objects of Bremen inner city using DBSCAN( $MinPts=2$ ,  $Eps=300$ ).

relations between *cluster pairs*. The output of the equation 4.1 indicates the percentage of spatial relations that can be saved with respect to the original size of  $\mathcal{G}_D$ .

In Figure 4.10, we depict three cases for the qualitative data reduction by DBSCAN, in which 12 database objects are clustered. For simplicity, we assume that the *disjoint* spatial relations between cluster pairs are inferable and allow for reduction. For instance, if we know that two clusters are *disjoint* from each other, we can directly conclude that all pairwise objects of those clusters are *disjoint* as well. Thus, there is no need to compute and store the *disjoint* relations among the pairwise objects of clusters. In the first case (Figure 4.10(a)) all objects are included in a single cluster which means that no reduction is possible, as the spatial relations between all object pairs need to be abstracted. In Figure 4.10.b, the database objects are grouped into two clusters. Accordingly, the reduction rate based on the equation previously mentioned can be computed as follows:

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---



**Figure 4.10:** Examples of qualitative data clustering and reduction by DBSCAN: (a) all objects are grouped into a single cluster, (b) all objects are grouped into two clusters, and (c) all objects are grouped into four clusters.

$$rate(\mathcal{G}_{\mathcal{D}}) = \left[ 1 - \frac{(((6*6)-6)+((6*6)-6)+(2))}{((12*12)-12)} \right] \cdot 100 \approx 53 \text{ \%}.$$

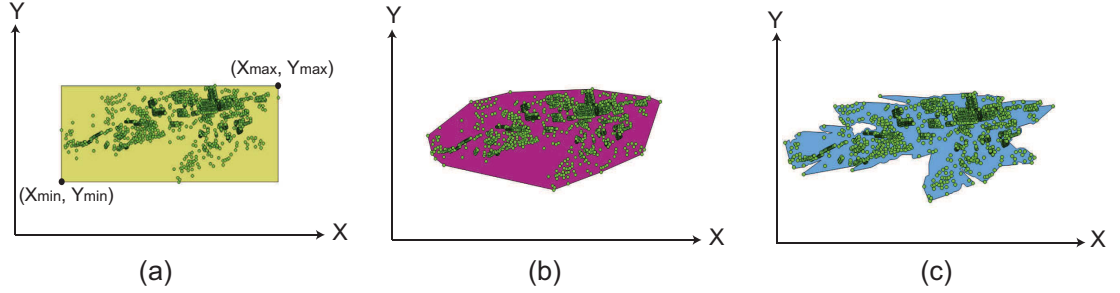
Similarly, we can calculate the reduction rate in the third case (Figure 4.10(b)) to be approximately 73 %. From the aforementioned cases, it is apparent that changing the number of clusters has an impact on the qualitative data reduction rates.

##### 4.4.1.1 Enclosing Clusters

The DBSCAN clustering presented previously identifies points that are near to each other as clusters by giving them labels (e.g., all points in cluster<sub>1</sub> ( $C_1$ ) can be labelled as 1 and so forth) that indicate to which cluster they belong. However, in order to abstract QSRs between clusters, point clusters must be first transformed (or approximated) into regions. More precisely, given the set of points which constitute cluster  $C_i$ ,  $C_i$  needs to be transformed into a region by capturing its approximate shape. Some methods for achieving this result are: Minimum Bounding Rectangle (MBR) (Buckley, 2008), Convex Hull (CH) (Chan, 1996), and ConCave Hull (CCH) (Duckham et al., 2008). These are the most popular methods, and we will consider them in this dissertation to capture the shape of  $C_i$ .

##### *Minimum Bounding Rectangle:*

Given a cluster  $C_i$  in a 2D-space, the MBR is the axis-aligned minimum bounding rectangle whose edges are parallel to the coordinate axes. All points of cluster  $C_i$  fall within this rectangle (see Figure 4.11(a)). The minimum and maximum extents of the rectangle are specified by two coordinate pairs:  $(X_{min}, Y_{min})$  and  $(X_{max}, Y_{max})$ ,



**Figure 4.11:** An example: the MBR, CH, and CCH of a cluster.

where  $X_{min}$  denotes the X-axis minimum value of  $S$ ,  $Y_{min}$  denotes the Y-axis minimum value of  $C_i$ ,  $X_{max}$  denotes the X-axis maximum value of  $C_i$ , and  $Y_{max}$  denotes the Y-axis maximum value of  $C_i$ . The time complexity of finding MBR is  $O(|C_i|)$ , since all the points of  $C_i$  need to be tested to determine the two endpoints of its major diagonal.

#### *Convex Hull:*

Given a cluster  $C_i$  in a 2D-space, a Convex Hull denoted by  $CH(C_i)$  is the set of the smallest convex polygon containing all points of  $C_i$  (see Figure 4.11(b)). In particular, the  $CH(C_i)$  is derived by intersecting all the possible convex sets that enclose the points in  $C_i$  (Chan, 1996). Chan (1996) proposes an algorithm that only requires  $O(|C_i| \log(h))$  steps for computing  $CH(C_i)$  in two and three dimensions, where  $h$  is the number of points on the boundary of the CH.

#### *Concave Hull:*

ConCave Hull, denoted by  $CCH(C_i)$ , aims to characterize the shape of  $C_i$  that represents the area occupied by all points in  $C_i$  by generating convex and non-convex hull polygons (see Figure 4.11(c)) (Edelsbrunner et al., 1983). It usually takes a single parameter  $k_{percent}$ , that can be adjusted to determine the level of smoothness of the computed polygon. In turn, Duckham et al. (2008) propose an algorithm that requires  $O(|C_i| \log|C_i|)$  steps to computing the  $CCH(C_i)$ ; it is based on the Delaunay triangulation of the points.

## 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

### 4.4.1.2 Qualitative Spatial Relations Between Clusters

The main goal of computing the QSRs between the cluster pairs is to identify the QSRs that can be inferred from each other. In this dissertation, the inferable QSRs are referred to as *decisive* relations and the non-inferable ones as *indecisive* relations.

**Definition 13** (A *decisive* spatial base relation).

Let  $\mathcal{B}$  be a Jointly Exhaustive and Pairwise Disjoint (JEPD) set of binary base relations defined over a domain  $D$ , and let  $R \in 2^{\mathcal{B}}$  be a spatial (disjunctive) relation. Then a spatial relation  $R$  between any two shapes<sup>1</sup> of clusters ( $\text{shape}(C_i)$ ,  $\text{shape}(C_j)$ ) is said to be *decisive* if and only if

$$(\text{shape}(C_i), \text{shape}(C_j)) \in R \implies \forall c_i \in C_i, \forall c_j \in C_j : (c_i, c_j) \in R.$$

Below, the following qualitative aspects are addressed separately in order to exploit the capability of reducing the QSRs: (1) topology, (2) distance, and (3) direction.

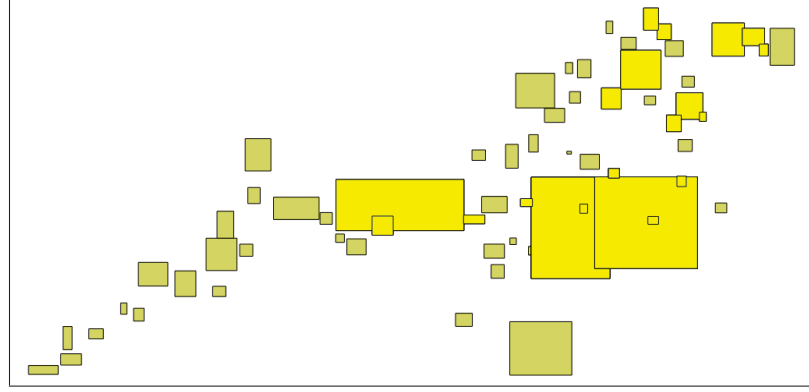
#### Topology:

In order to compute the precise topological relations between clusters, the shape of the clusters must be captured as tightly as possible. To achieve this result, we use ConCave Hull (CCH), because of its ability to produce tighter polygons than the CH and MBR. Recall that the topological model we introduced in Section 2.2.1 has eight topological relations, which can be abstracted between simple convex regions: *equal*, *disjoint*, *meets*, *overlaps*, *contains*, *covers*, *inside*, and *coveredBy*. Here, we only focus on *disjoint* relations, since a direct conclusion can be drawn regarding the disjointedness of clusters. Therefore, even if clusters contain holes, the correct *disjoint* relation can be captured between these clusters. In this dissertation, we argue that the CCH generates fewer non-*disjoint* relations than the MBR and CH, which means that the CCH can achieve a higher reduction rate than the others. Figure 4.12 illustrates three examples, which show that the CCH produces the fewest number of non-*disjoint* relations, thereby achieving a higher topological reduction rate than the others.

In (Egenhofer and Franzosa, 1991), two clusters  $C_i$  and  $C_j$  are called *disjoint* if the intersection of  $C_i$ 's interior, boundary, and exterior with  $C_j$ 's interior, boundary, and exterior is *empty*. Since, the disjointness of the concave hulls of two clusters  $\text{CCH}(C_i)$

---

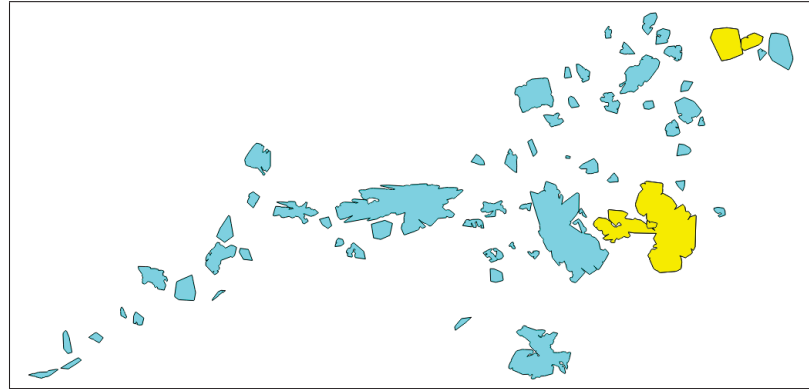
<sup>1</sup>Such as MBR, CH, or CCH cf. Section 4.4.1.1.



(a) the MBR generates more non-*disjoint* relations than the CH and CCH.



(b) the CH generates fewer non-*disjoint* relations than the MBR.



(c) the CCH generates fewer non-*disjoint* relations than both the MBR and CH.

**Figure 4.12:** An example: computing *disjoint* relations between clusters based on three cluster representations the MBR, CH, and CCH, where the yellow color presents the non-*disjoint* relations.

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

and  $\text{CCH}(C_j)$ , implies that all the points of  $\text{CCH}(C_i)$  are *disjoint* from all the points of the  $\text{CCH}(C_j)$ , we therefore consider *disjoint* as a *decisive* relation.

##### Distance:

The absolute distance model presented in Section 4.2 distinguishes the four qualitative spatial relations: *ZeroDist*, *near*, *medium*, and *far*. In order to compute a distance between clusters, we use the MBRs as suitable representations of clusters for two reasons: (1) the maximum and minimum distances can be computed based on the two endpoints of the MBRs and (2) measuring the distance between the MBRs is computationally inexpensive.

Let  $p = (p_1, p_2)$  to be a 2D point, where  $p_1$  and  $p_2$  are its coordinates. Also, let  $R$  be a 2D MBR defined by the two endpoints of its major diagonal  $r = (r_1, r_2)$  and  $r' = (r'_1, r'_2)$ , where  $r'_1 > r_1$ ,  $r'_2 > r_2$ ,  $(r_1, r_2)$ , and  $(r'_1, r'_2)$  are the coordinates of  $r$  and  $r'$  respectively. Similarly, let  $S$  be a 2D MBR defined by the two endpoints of its major diagonal  $s = (s_1, s_2)$  and  $s' = (s'_1, s'_2)$ , where  $s'_1 > s_1$ ,  $s'_2 > s_2$ ,  $(s_1, s_2)$ , and  $(s'_1, s'_2)$  are the coordinates of  $s$  and  $s'$  respectively.

Then the minimum and maximum distances, MINDIST and MAXDIST respectively between any two rectangles and between a point and a rectangle, are given by (Ramaswamy et al., 2000; Roussopoulos et al., 1995):

**Definition 14** ( $\text{MINDIST}(p, R) = \sqrt{\sum_{i=1}^2 x_i^2}$ , where).

$$x_i = \begin{cases} r_i - p_i & \text{if } p_i < r_i; \\ p_i - r'_i & \text{if } r'_i < p_i; \\ 0 & \text{otherwise.} \end{cases}$$

**Definition 15** ( $\text{MINDIST}(R, S) = \sqrt{\sum_{i=1}^2 x_i^2}$ , where).

$$x_i = \begin{cases} r_i - s'_i & \text{if } s'_i < r_i; \\ s_i - r'_i & \text{if } r'_i < s_i; \\ 0 & \text{otherwise.} \end{cases}$$

**Definition 16** ( $MAXDIST(p, R) = \sqrt{\sum_{i=1}^2 x_i^2}$ , where).

$$x_i = \begin{cases} r'_i - p_i & \text{if } p_i < \frac{r_i + r'_i}{2}; \\ p_i - r_i & \text{otherwise.} \end{cases}$$

**Definition 17** ( $MAXDIST(R, S) = \sqrt{\sum_{i=1}^2 x_i^2}$ , where).

$$x_i = \left\{ \max(|s'_i - r_i|, |r'_i - s_i|) \right\}.$$

Based on our distance model, the relation between the MBRs of clusters  $R$  and  $S$  is considered *ZeroDist* if  $MINDIST(R, S) = 0$ , *near* if  $d_0 < MAXDIST(R, S) \leq d_1$ , *medium* if  $d_1 < MINDIST(R, S) \leq d_2 \wedge d_1 < MAXDIST(R, S) \leq d_2$ , *far* if  $d_2 < MINDIST(R, S) < d_3$ , and *indecisive* if otherwise.

Table 4.1 lists the *decisive* and *indecisive* relations based on the derived relations of the MAXDIST and the MINDIST functions. If a distance relation between  $R$  and  $S$

**Table 4.1:** The distance decisive and indecisive relations.

$MINDIST(R, S) \backslash MAXDIST(R, S)$	$ZeroDist$	$near$	$medium$	$far$
$ZeroDist$	<i>indecisive</i>	-	-	-
$near$	<i>indecisive</i>	<i>decisive</i>	-	-
$medium$	<i>indecisive</i>	<i>indecisive</i>	<i>decisive</i>	-
$far$	<i>indecisive</i>	<i>indecisive</i>	<i>indecisive</i>	<i>decisive</i>

is considered *indecisive*, distance relations are computed over all pairs of points  $(r, s)$ , where  $r \in R$  and  $s \in S$ , since the pairs of points of two clusters do not have a unique distance relation. For example, a distance relation between points pair  $r_1 \in R$  and  $s_1 \in S$  could be *near*, *medium* for points  $r_2 \in R$  and  $s_2 \in S$  of the same MBRs of  $R$  and  $S$ . Unfortunately *indecisive* relations have a high space and time complexity. To deal with this issue, clusters can be divided into smaller sub-clusters that have *decisive* distance relations.

## 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

### Direction:

The Cardinal Direction Model (CDM) for extended objects and the cone-based model in Section 2.2.2 contain nine base (or ‘single-tile’) relations can be differentiated. Since the CDM partitions the space based on the MBR of a reference object, a directional relation between clusters can be computed based on their MBRs as well. In order to guarantee that a cluster covers the actual extent of objects, a MBR of a cluster is derived by computing the MBR of the MBRs of objects occurring in this cluster. In the CDM, based on the transitivity property of the relation, we consider the four directional relations as *decisive*, as they are transitive: **NE**, **NW**, **SE**, and **SW** (see Lemma 1).

**Lemma 1.** *The directional relations **NE**, **NW**, **SE**, and **SW** are decisive.*

### Proof.

Given three points shown in Figure 4.13(a),  $c_1 = (x_1, y_1)$ ,  $c_2 = (x_2, y_2)$ , and  $c_3 = (x_3, y_3)$ , where a point  $c_1 \in \text{MBR}(C_i)$ , the most **NE** point  $c_2 \in \text{MBR}(C_i)$  and a point  $c_3 \in \text{MBR}(C_j)$ . Now  $c_2 \text{NE} c_1 \iff x_2 > x_1 \wedge y_2 > y_1$  and  $c_3 \text{NE} c_2 \iff x_3 > x_2 \wedge y_3 > y_2 \implies x_3 > x_1 \wedge y_3 > y_1 \implies c_3 \text{NE} c_1$ . Similarly, the transitivity of the other three relation can be proven.

In contrast, we consider **N**, **S**, **E**, and **W** as well as multi-tile relations as *indecisive*, since they are non-transitive (Lemma 2).

**Lemma 2.** ***N**, **S**, **E**, and **W** as well as multi-tile relations are indecisive.*

### Proof.

Given three points shown in Figure 4.13(b),  $c_1 = (x_1, y_1)$ ,  $c_2 = (x_2, y_2)$ , and  $c_3 = (x_3, y_3)$ , where a point  $c_1 \in \text{MBR}(C_i)$ , the most **N** point  $c_2 \in \text{MBR}(C_i)$  and a point  $c_3 \in \text{MBR}(C_j)$ . Assume that  $c_3 \text{N} c_1$ . Now  $c_2 \text{N} c_1 \iff x_2 = x_1 \wedge y_2 > y_1$ , but  $c_3 \text{NW} c_2 \iff x_3 < x_2 \wedge y_3 > y_2 \not\Rightarrow x_3 < x_1 \wedge y_3 > y_1 \not\Rightarrow c_3 \text{N} c_1$ . Similarly, the non-transitivity of the other relations as well as multi-tile relations can be proven.

As we have shown that the *indecisive* are non-transitive, the directional relations of the cone-based model are thusly *indecisive*, since they are non-transitive.

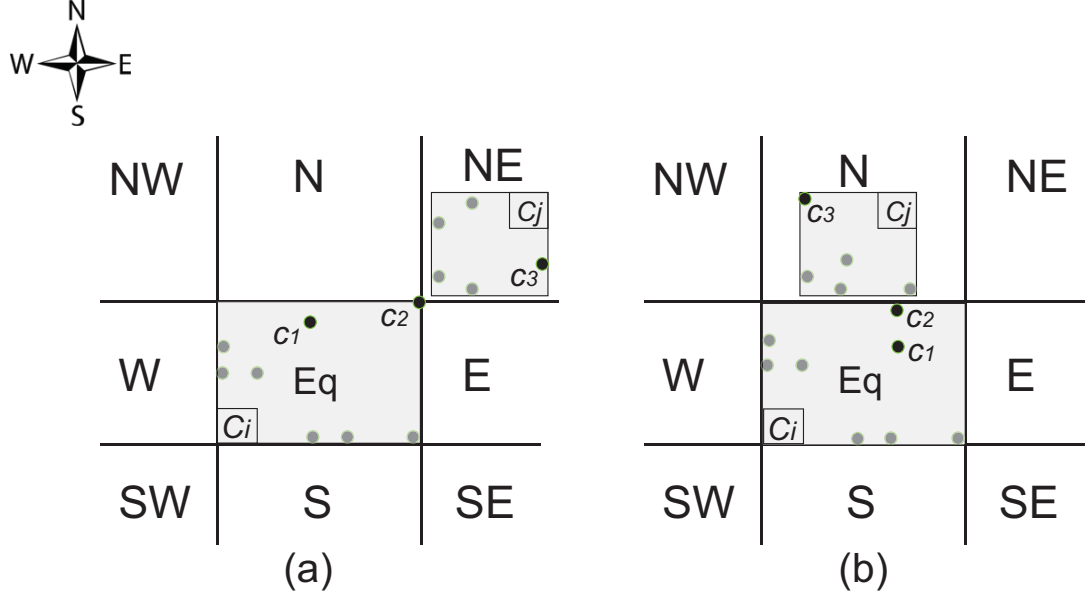
### 4.4.1.3 Abstracting a Qualitative Spatial Layer From Clusters

Algorithm 3 is an algorithm for abstracting QSRs within clusters and between cluster pairs. First, it iterates over each cluster, abstracts QSRs within each cluster by using the *AbstractDataBaseGraph* procedure<sup>1</sup>, and eventually stores them in  $\mathcal{G}_{\mathcal{D}}^C$  (lines 2 to 4).

---

<sup>1</sup>This procedure is already mentioned in Section 4.3 and used to abstract QSRs.





**Figure 4.13:** (a) shows the transitive relation  $NE$  between the reference cluster  $C_i$  and the primary cluster  $C_j$ , (b) shows the non-transitive relation  $N$  between the reference cluster  $C_i$  and the primary cluster  $C_j$ .

Secondly, it iterates over all pairwise *disjoint* clusters in  $C$  and computes the relation  $r$  (lines 5 to 8). Finally, if  $r$  is *decisive* then the relation, the edges, and their labels are added to  $C^R$  (lines 9 and 10). Otherwise, all pairwise *disjoint* object pairs of two clusters are iterated over and the relations are computed and added to  $\mathcal{G}_D^C$  (lines 11 to 22).

#### 4.4.1.4 Matching the Qualitative Spatial Layer of Clusters

In this sub-section, we present an algorithm called DBSCAN Matcher (DM), which aims to match  $\mathcal{G}_Q$  against the graph databases of clusters. Algorithm 4 first matches  $\mathcal{G}_Q$  against each graph database of cluster in  $\mathcal{G}_D^C$  by calling the `Qualitative_Layer_Matcher()` function (lines 2 to 4). We note that `Qualitative_Layer_Matcher()` is already presented in Section 4.3.2 and used to match  $\mathcal{G}_Q$  against the qualitative spatial layer. Afterwards, it checks each spatial relation of  $\mathcal{G}_Q$ , and determines whether it is a *decisive* relation or not (lines 5 to 7). If relation is determined as *decisive*, it detects the cluster pairs holding the relation. In the end, it runs through all pairwise *disjoint* clusters in  $C$  and extracts the object pairs that have the same labels as the ones of  $\mathcal{G}_Q$  (lines 8 to 11).

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---



---

**Algorithm 3:** Abstract\_Relations\_Clusters(*Clusters C, GeomtryClusters F*)

---

**input** :  $C$ : clusters and their database objects  
**output**:  $\mathcal{G}_D^C$ : a set of complete graphs of the clusters DB that contains the object pairs and their corresponding relations,  $C^R$ : a complete graph of the clusters that contains the clusters pairs and their corresponding relations

```

1 initialization:  $r \leftarrow \text{NULL}$ ;  $rc \leftarrow \text{NULL}$ ;  $\mathcal{G}_D \leftarrow \text{NULL}$ ;  $SC \leftarrow \text{NULL}$ ;  $FC \leftarrow \text{NULL}$ ;
2 for  $k \leftarrow 1$  to  $|C|$  do
3    $\mathcal{G}_D^C.\text{add} \leftarrow \text{AbstractDataBaseGraph}(C_k, F_k)$ ;
4 end
5 for  $i \leftarrow 1$  to  $|C|$  do
6   for  $j \leftarrow 1$  to  $|C|$  do
7     if  $(C_i.\text{id} \neq C_j.\text{id})$  then
8        $r \leftarrow \text{ComputeQSRels}(\text{GetShap}(F_i), \text{GetShap}(F_j))$ ;
9       if  $r$  is decisive then
10         $C^R.\text{add} \leftarrow (C_i, r, C_j, C_i.\text{id}, C_j.\text{id})$ ;
11      else
12         $FC \leftarrow C_i$ ;
13         $SC \leftarrow C_j$ ;
14        for  $v \leftarrow 1$  to  $|FC|$  do
15          for  $e \leftarrow 1$  to  $|SC|$  do
16            if  $(FC_v.\text{id} \neq SC_e.\text{id})$  then
17               $rc \leftarrow \text{ComputeQSRels}(F_v, F_e)$ ;
18               $\mathcal{G}_D^C.\text{add} \leftarrow (FC_v, rc, SC_e, FC_v.\text{id}, SC_e.\text{id})$ ;
19            end
20          end
21        end
22      end
23    end
24  end
25 end

```

---

---

**Algorithm 4:** DBSCAN\_Matcher( $\mathcal{G}_{\mathcal{D}}^C, C^R$ , Clusters  $C$ , Query  $\mathcal{G}_Q$ )
 

---

**output:**  $\mathcal{M}$ : a set of matches satisfying  $\mathcal{G}_Q$   
 1 initialization:  $\mathcal{R} \leftarrow \text{NULL}$ ;  $\mathcal{M} \leftarrow \text{NULL}$ ;  
 2 **for**  $k \leftarrow 1$  **to**  $|C|$  **do**  
 3      $\mathcal{M}.\text{add} \leftarrow \text{Qualitative\_Layer\_Matcher}(\mathcal{G}_{\mathcal{D}k}^C, \mathcal{G}_Q)$ ;  
 4 **end**  
 5  $\mathcal{R} \leftarrow \mathcal{G}_Q.\text{GetRelations}()$ ;  
 6 **for**  $t \leftarrow 0$  **to**  $|\mathcal{R}|$  **do**  
 7     **if**  $\mathcal{R}_t$  *is decisive* **then**  
 8          $P \leftarrow C^R.\text{GetPairs}(\mathcal{R}_t)$ ;  
 9         **foreach**  $(p_i, p_j) \in P \wedge i \neq j$  **do**  
 10              $\mathcal{M}.\text{add} \leftarrow C.\text{GetMatches}(C_i, C_j, p_i, p_j)$ ;  
 11         **end**  
 12     **end**  
 13 **end**

---

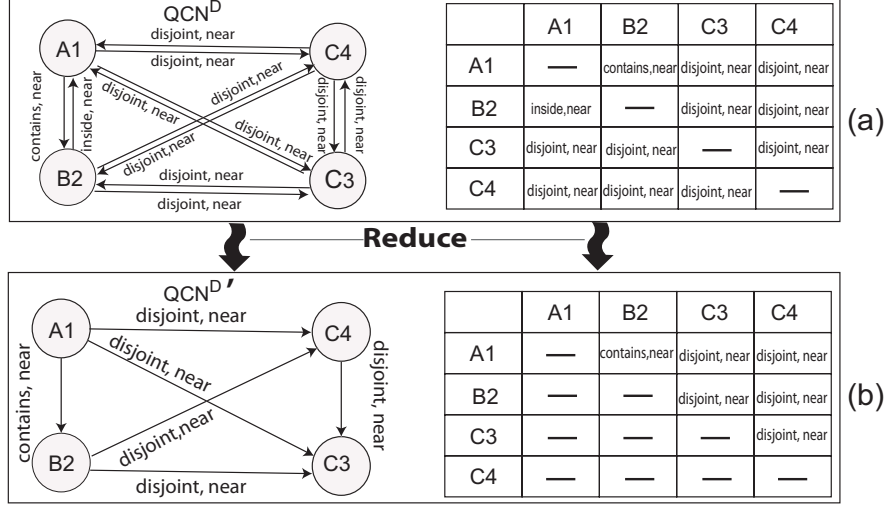
#### 4.4.2 Qualitative Data Reduction By a Converse Operation

In the previous section, we have applied a clustering approach to reduce the size of the graph database ( $\mathcal{G}_{\mathcal{D}}$ ). Here, we apply a converse operation provided by qualitative spatial models (cf. Section 2.1.1) to reduce the size of  $\mathcal{G}_{\mathcal{D}}$  as well. Based on the converse property of qualitative models, binary relations can be permanently deleted from  $\mathcal{G}_{\mathcal{D}}$ .

$\mathcal{G}_{\mathcal{D}}$  can be represented by a multi-Qualitative Constraint Network ( $\text{QCN}^D$ ) which allows for performing converse and composition operations. Given qualitative models that have a unique converse property, a symmetric graph of  $\text{QCNs}^D$  can be exploited. Hence, only half size of the  $\text{QCN}^D$  needs to be considered in a query processing, whereas the other half can be safely pruned. Figure 4.14.(a) shows an example of a  $\text{QCN}^D$  containing four objects ( $A_1, B_2, C_3, C_4$ ) and their binary distance and topology relations. It also shows how a  $\text{QCN}^D$  can be represented by a symmetric 2D matrix. Figure 4.14.(b) shows that half of the  $\text{QCN}^D$  can be pruned based on symmetry, which results in the new labelled graph  $\text{QCN}^{D'}$ .

Table 4.2 shows the labels of inverse relations for the eight binary relations of the 9-Intersection Model (Egenhofer and Franzosa, 1995). It also illustrates that the last four labels of relations and their inverses are not the same. For instance, the label of

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION



**Figure 4.14:** An example: pruning half of  $QCND$  space based on symmetry.

**Table 4.2:** Eight binary relations of 9-Intersection-Model and their inverses from (Egenhofer, 1994a).

Nr	Relation $R$	Inverse of Relation $R^\sim$
1	$disjoint(A, B)$	$disjoint(B, A)$
2	$meet(A, B)$	$meet(B, A)$
3	$equal(A, B)$	$equal(B, A)$
4	$overlap(A, B)$	$overlap(B, A)$
5	$inside(A, B)$	$contains(B, A)$
6	$contains(A, B)$	$inside(B, A)$
7	$covers(A, B)$	$coveredBy(B, A)$
8	$coveredBy(A, B)$	$covers(B, A)$

the topological relation *inside* is the inverse of the *contains* relation, and vice versa.

Similarly, a spatial query can be given as a multi-Qualitative Constraint Network ( $QCND$ ). If the  $QCND$  is given as a complete graph, half of the binary relations of a spatial query can be pruned based on a symmetric property as well. However, the binary relations of a spatial query must be checked to ensure that they belong to the non-pruned part of the  $QCND$ . For instance, when  $QCND$  contains binary relations that belong to the pruned part of  $QCND$ , the converse operation is applied on those binary relations.

**Example 1** (Converse operation over binary relation of a spatial query).

*In a query “Find me a park that contains a restaurant”, the topological relation contains does not belong to  $QCN^D$ . Hence, the converse operation must be applied on the contains relation to be the inside relation and the order of object pairs must be reversed. Consequently, the spatial query changes to be “Find me a restaurant that is inside a park”.*

#### 4. QUERYING, REDUCING, AND MATCHING QUALITATIVE INFORMATION

---

**Table 4.3:** Comparison between the three matching approaches.

Approach	Space Complexity	Time Complexity	QSL	Computing Relations
<b>naive</b>	$O(n)$	$O(n^2)$	Not required	Run time
QLM	$O(n^2)$	$O(n^2)$	Required	In advance
DM	$O( \mathcal{G}_D^C ) + O( C^R )$	$O( \mathcal{G}_D^C ) + O( C^R )$	Partially required	Partially at run time

#### 4.5 Summary

There is still a fairly wide gap between the qualitative concepts of a human and the quantitative data stored in spatial databases of Geographic Information Systems (GISs). Accordingly, we have integrated the qualitative spatial models into the geometry object model to enable the intuitive and qualitative formalism of queries in GISs. To avoid the extra costs of query processing, we have abstracted the Qualitative Spatial Layer (QSL) that covers the spatial aspects of space from spatial databases. Since the space demands of the QSL are high, we have applied two qualitative data reduction strategies. This chapter has emphasized on the three matching approaches for answering QSQs: (1) a **naive approach**, (2) Qualitative Layer Matcher (QLM), and (3) DBSCAN Matcher (DM). The properties and capabilities vary from one approach to the next. In Table 4.3, we summarize these properties and capabilities for the approaches. Aside from space and time complexity, Table 4.3 shows the QSL and computational requirements for each approach, where  $n$  is the number of database objects,  $|\mathcal{G}_D^C|$  is the number of spatial relations within clusters and between pairwise objects of cluster pairs holding *indecisive* relations, and  $|C^R|$  is the number of spatial relations between cluster pairs holding *decisive* relations. From Table 4.3 it is apparent that the **naive approach** does not require the QSL, and thus  $O(n)$  space is required. However, the approach needs to compute  $O(n^2)$  relations from database geometries at *run time*, which is computationally expensive. Conversely, QLM requires the QSL, which results in  $O(n^2)$  space, but it does not need to compute spatial relations at run time. DM acts as a middle case, in which some of spatial relations need to be computed from database geometries.

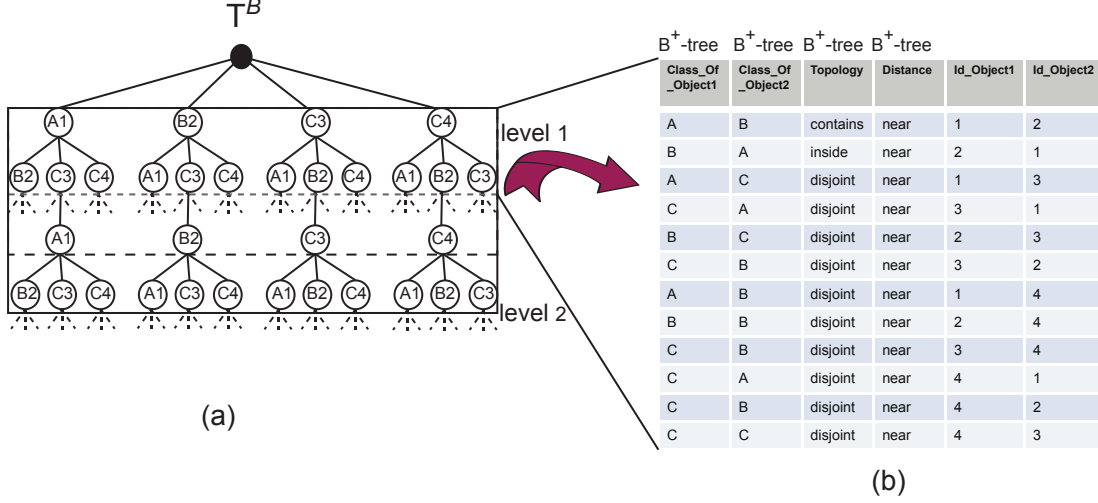
# Optimizing Indexing Approaches for Spatial Databases

In Chapter 4 we pointed out that the interpretation tree (*Itree*) is an adequate representation for solving a sub-graph isomorphism matching problem. Additionally, we used Breadth-First Search (BFS) to match Qualitative Spatial Queries (QSQs) against the entries of *Itree*. However, processing QSQs using BFS has an exponential space and time complexity. In order to reduce the complexity of processing QSQs, we propose five optimization indexing approaches: (1) A Hybrid Interpretation Tree and  $B^+$ -Tree (HITBT) (Section 5.1), (2) Qualitative Hash Table Indexing (QHTI) (Section 5.2), (3) Qualitative Hash Table Compression (QHTC) (Section 5.3), (4) QHTC of Qualitative Models (QHTC<sup>M</sup>) (Section 5.4), and (5) QHTC of Object Pairs (QHTC<sup>P</sup>) (Section 5.5).

## 5.1 A Hybrid Interpretation Tree and $B^+$ -Tree

In Section 4.3.2 we indicated that the graph database  $\mathcal{G}_{\mathcal{D}}$  can be decomposed into *Itree* at run time to be matched by a graph query  $\mathcal{G}_Q$  via BFS. However, the central disadvantage of this approach is that all entries of *Itree* must be traversed (in an arbitrary manner). This procedure requires an exponential number of steps to find all possible answers to a  $\mathcal{G}_Q$ . Several approaches are proposed to decompose graph databases into trees in advance, e.g., (Bodlaender, 1997; Matoušek and Thomas, 1991; Robertson and Seymour, 1986). In particular, the authors argue that the decomposition of a graph with a limited tree-width allows for solving the sub-graph matching problem

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES



**Figure 5.1:** An example: the index construction of the first level of  $T^B$  of HITBT.

in polynomial time. The graph  $\mathcal{G}_{\mathcal{D}}$  with  $k$ -width can be decomposed into  $k$ -trees, where  $k$ -trees enumerate and store all possible matchings in all the  $k$ -levels of  $\mathcal{G}_{\mathcal{D}}$  in advance. This approach can be actually viewed as pre-constructed *Itree* of  $\mathcal{G}_{\mathcal{D}}$  with  $k$ -width. Accordingly, the construction of *Itree* of  $\mathcal{G}_{\mathcal{D}}$  with  $k$ -width can be done in polynomial time (Bodlaender, 1997). This procedure considerably reduces time complexity of sub-graph isomorphism matching, while it brings an exponential space complexity. In this chapter, we assume that the queries contain a limited number of binary qualitative spatial relations. Simplified, we refer to qualitative spatial relations as relations and for QSQs as queries or  $\mathcal{G}_{\mathcal{Q}}(s)$ . Moreover, we focus on the precomputed *Itree* in which we reduce space and time complexity of processing the queries.

Constructing *Itree* in advance allows for constructing B<sup>+</sup>-trees (see Section 3.1.2.1 for more details about B<sup>+</sup>-trees) index on its entries which implies reduction of query processing time. In order to reduce the processing time of the queries, we propose a Hybrid Interpretation Tree and B<sup>+</sup>-Tree (HITBT) approach. HITBT offers three operations: Index Construction-HITBT, Search-HITBT, and Delete-HITBT.

### 5.1.1 Index Construction-HITBT

As shown in Figure 5.1(a), the Index Construction (IC) of HITBT can be done by decomposing  $\mathcal{G}_{\mathcal{D}}$  into several levels of tree  $T^B$ , in which B<sup>+</sup>-trees are deployed on each level of  $T^B$ . In particular, Figure 5.1(b) illustrates that B<sup>+</sup>-trees are deployed on the



---

## 5.1 A Hybrid Interpretation Tree and B<sup>+</sup>-Tree

---

labels of object pairs and their relations of the first level of  $T^B$ . In this dissertation, the  $\mathcal{G}_D$  dimensions denote the labels of spatial relations holding object pairs and the  $\mathcal{G}_D$  attributes denote the labels of object pairs and their corresponding dimensions. Algorithm 5 lists the steps to construct  $T^B$  from  $\mathcal{G}_D$ . With  $\mathcal{G}[g]$ , we denote the  $g^{th}$  vertex of a graph  $\mathcal{G}_D$ . The algorithm iterates all possible  $k$ -ary tuples of objects with  $k \geq 2$  and  $k < \ell$ , where  $\ell$  is the maximum tree-width<sup>1</sup> (line 3). For instance,  $k = 2$  constructs all graph sets that consist of two object pairs and their relations. Subsequently, the labels of all object pairs and their relations per level  $k$  are stored in a temporal tree (*TempTG*) (lines 3 to 9). In the end, *TempTG* is added into tree  $T^B$  and then B<sup>+</sup>-trees are deployed on the attributes of  $T^B$  (lines 10 and 11).

---

**Algorithm 5:** IndexConstruction\_HITBT(*DBgraph*  $\mathcal{G}_D$ , *TreeLevel*  $\ell$ )

---

**output:**  $T^B$ : all possible sets of object pairs and their relations indexed by B<sup>+</sup>-trees

```

1 initialization:  $TempTG \leftarrow \text{NULL}$ ;  $T^B \leftarrow \text{NULL}$ ;  $\ell \leftarrow \text{MaxQuerySize}$ ;
2  $T^B.\text{Level}(1) \leftarrow \mathcal{G}_D$ ;
3 for  $k \leftarrow 2$  to  $\ell$  do
4    $TempHG \leftarrow \text{NULL}$ ;
5   for  $h \leftarrow 1$  to  $\ell$  do
6     for  $m \leftarrow 1$  to  $|T^B.\text{Level}(k-1)|$  do
7        $TempHG.\text{Add}(\mathcal{G}_D[h], T^B.\text{Level}(k-1)[m]);$ 
8     end
9   end
10   $T^B.\text{Add}(TempHG, k)$ ;
11   $T^B.\text{Level}(k).\text{Apply.B}^+\text{-trees}()$ ;
12 end
```

---

### 5.1.2 Search-HITBT

In order to match a query, HITBT matches each attribute of  $T^B$  by each attribute of  $\mathcal{G}_Q$  using the search operation of B<sup>+</sup>-trees (cf. Section 3.1.2.1). Eventually, the returned results of the attributes are intersected to obtain the final results of the query. Algorithm 6 shows the steps to match  $\mathcal{G}_Q$  against the entries of  $T^B$ . Firstly, the level of  $T^B$  that

---

<sup>1</sup>In our case the maximum tree-width is restricted to a maximum size that a query is allowed to be.

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---

needs to be searched is identified via the order of  $\mathcal{G}_Q$  (line 2). Next, the attributes of  $\mathcal{G}_Q$  are matched against the *attributes* of  $T^B$  using  $B^+$ -trees search operation (line 3). Lastly, the results of the *attributes* are intersected with the values of new *attributes* and then stored in  $\mathcal{M}$  (lines 4 to 7).

---

**Algorithm 6:** Match\_HITBT( $B^+$ -trees  $T^B$ , *QueryGraph*  $\mathcal{G}_Q$ )

---

**output :**  $\mathcal{M}$ : a set of matches satisfying  $\mathcal{G}_Q$

---

```

1 initialization:  $\mathcal{M} \leftarrow \text{NULL}$ ;  $FirstAttr \leftarrow \text{NULL}$ ;  $T_{level} \leftarrow \text{NULL}$ ;
2  $T_{level} \leftarrow T^B.\text{Level}(|\mathcal{G}_Q|)$ ;
3  $FirstAttr \leftarrow \mathcal{G}_Q.\text{Get\_First\_Attribute}()$ ;
4  $\mathcal{M} \leftarrow T_{level}.FirstAttr.B^+\text{-trees}.\text{Search}(FirstAttr)$ ;
5 foreach attributes  $attr \in \mathcal{G}_Q$  with  $attr \neq FirstAttr$  do
6    $\mathcal{M} \leftarrow \mathcal{M} \cap T^B.attr.B^+\text{-trees}.\text{Search}(\mathcal{G}_Q.attr)$ ;
7 end
```

---

### 5.1.3 Delete-HITBT

Given an object  $o$  that is requested to be deleted, first HITBT needs to find all the entries of each level of  $T^B$  that match label  $o$ . Accordingly, the matched entries are deleted from  $T^B$  and result in a new tree  $T^{B'}$ . Algorithm 7 outlines the steps to delete  $o$  from  $T^B$ . First, in every level of  $T^B$  with  $k \geq 1$  all object pairs that contain the object  $o$  are fetched through  $B^+$ -trees search operation and then stored in a temporal variable  $TMP$  (lines 2 and 3). Subsequently,  $TMP$  entries are iterated where object pairs that contain the exact *Id* of  $o$  are deleted from  $T^B$  and  $B^+$ -trees (lines 4 to 9). Finally, the structure of  $T^B$  and the pointers of  $B^+$ -trees are updated (lines 10 and 11).

### 5.1.4 Discussion

We have described the HITBT approach that aimed to accelerate the spatial query processing by deploying  $B^+$ -trees onto each level of  $T^B$  of HITBT. However, HITBT suffers from two major drawbacks: (1) high dimensionality and (2) high space complexity. The first drawback has been stemmed from applying  $B^+$ -trees on every attribute for each level of  $T^B$ . Due to this, a significant amount of space is needed to store  $B^+$ -trees indices. The second drawback has been caused due to the fact that the number of constructed sets of  $T^B$  is very highly grown with the number of the levels of  $T^B$ . Therefore, there is

---

**Algorithm 7:** Delete\_HITBT( $B^+$ tree  $T^B$ , Object  $o$ )

---

```

output:  $T^B$ :  $T^B$  without any occurrence of  $o$ 
1 initialization:  $TMP \leftarrow \text{NULL}$ ;
2 for  $k \leftarrow 1$  to  $T^B.\text{MaxLevel}$  do
3    $TMP \leftarrow T^B.\text{Level}(k).\text{GetPairs}(o)$ ;
4   foreach  $tmp \in TMP$  do
5     if  $tmp.\text{Contains}(o.Id)$  then
6        $T^B.\text{Level}(k).\text{Delete}(tmp)$ ;
7        $T^B.\text{Level}(k).B^+\text{-trees}.\text{Delete}(tmp)$ ;
8     end
9   end
10   $T^B.\text{Level}(k).\text{Update}()$ ;
11   $T^B.\text{Level}(k).B^+\text{-trees}.\text{Update}()$ ;
12 end

```

---

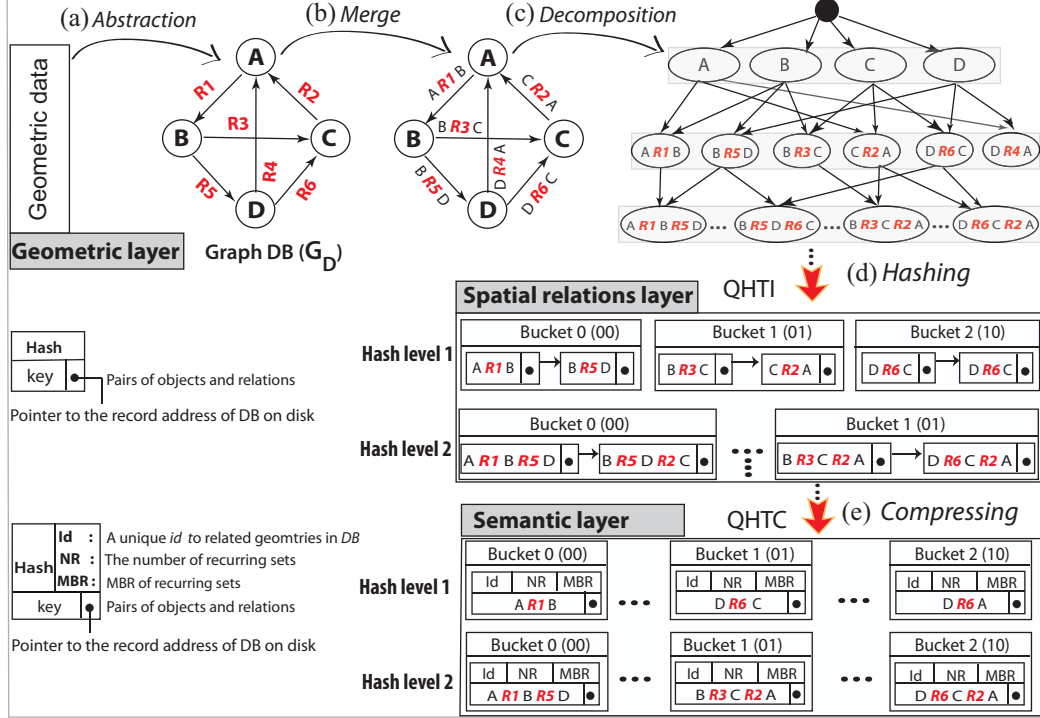
a necessity to develop approaches that extend and enhance the capability of HITBT by dealing with aforementioned drawbacks.

## 5.2 Qualitative Hash Table Indexing

In this section we introduce a Qualitative Hash Table Indexing (QHTI) to deal with a high dimensionality issue that has been explained in Section 5.1.4. We have sketched QHTI in (Al-Salman and Dylla, 2013). Here we detail QHTI and its architecture is depicted in Figure 5.2(a), (b), (c), and (d).

The objective of QHTI is to compute all qualitative spatial relations between object pairs in a database  $\mathcal{D}$  and store them in a hash table. QHTI particularly copes with the problem of high dimensionality by concatenating  $N$  attributes of the abstracted graph database ( $\mathcal{G}_{\mathcal{D}}$ ) into a single attribute. Again, the  $\mathcal{G}_{\mathcal{D}}$  dimensions denote the labels of spatial relations holding object pairs and the  $\mathcal{G}_{\mathcal{D}}$  attributes denote the labels of object pairs and their corresponding dimensions. Simplified, QHTI provides three operations: (1) Index Construction, (2) Search, and (3) Delete.

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

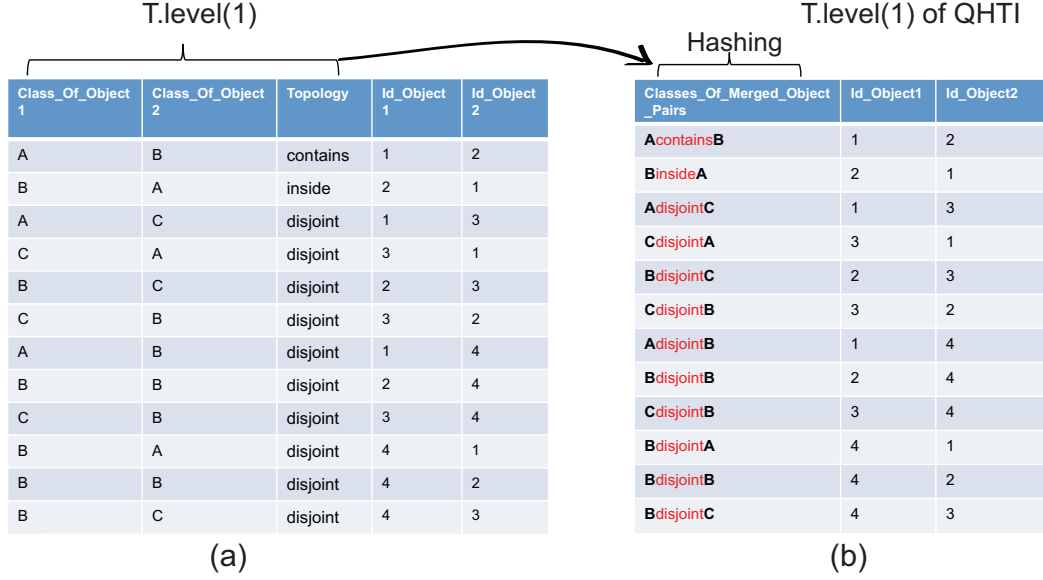


**Figure 5.2:** Architecture of Qualitative Hash Table Indexing (QHTI) and Compression (QHTC).

### 5.2.1 Index Construction-QHTI

QHTI index construction consists of two steps: (a) merging and (b) decomposition and hashing.

(1) *Merging*: In the merging step for each object pair in database  $\mathcal{D}$  and their corresponding edges specific labels are calculated from the quantitative values (Figure 5.2(a)). First, the qualitative relation  $r$  of a specific qualitative dimension is derived, e.g., topology (Egenhofer and Franzosa, 1995). The new label  $l$  results from joining the relation name with the names of the vertices, e.g., from the relation **Contains** holding between objects X and Y results in a single label “X\_Contains\_Y” (Figure 5.2(b)). So, we can derive a graph database  $\mathcal{G}_D$  by computing the relations from the geometric (or quantitative) data  $\forall o_i, o_j \in \mathcal{O}_D$  with  $o_i \neq o_j$ , generating the according label (merging), and adding them to  $\mathcal{G}_D$ . This procedure has a complexity of  $O(\eta|\mathcal{O}_D|^2)$ , where  $\eta$  is the number of qualitative dimensions.



**Figure 5.3:** An example: the index construction of the first level of T.

(2) *Decomposition and Hashing:* In the second step,  $\mathcal{G}_{\mathcal{D}}$  with  $k$ -width is decomposed into a tree T, where  $k \geq 2$ ,  $k < \ell$ , and  $\ell$  is the maximum tree-width<sup>1</sup>. Each level  $k$  of T comprises all possible subsets of  $k + 1$  vertices (i.e. objects) and their relations with (Figure 5.2(c) and (d)). We apply linear hashing (Litwin, 1980) to hash the entries in each level of the tree, because it is able to cope with databases that change their size dynamically (cf. Section 3.1.2.3). Hash keys and values are generated from object names and their relations (Figure 5.2(d)). We start with initializing the first tree level with  $\mathcal{G}_{\mathcal{D}}$  and iteratively build-up each tree level  $k$  by adding an object from  $\mathcal{G}_{\mathcal{D}}$  to the subsets derived in level  $k-1$  (Algorithm 8). With  $\mathcal{G}[g]$  ( $H[h]$ ) we denote the  $g^{th}$  vertex of a graph  $\mathcal{G}_{\mathcal{D}}$  ( $h^{th}$  object of a hash table  $H$ ). For each tree level linear hashing is applied.

Figure 5.3 gives an example how labels of the first level of T are generated (or merged) from the object names and their corresponding topological relations.

### 5.2.2 Search-QHTI

In order to match a query  $\mathcal{G}_{\mathcal{Q}}$  against the hash table entries, QHTI does not need to visit all hash entries. We only need to consider entries that have the same hash value as  $\mathcal{G}_{\mathcal{Q}}$ .

<sup>1</sup>Again, in our case the maximum tree-width is restricted to a maximum size that a query is allowed to be.

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---



---

### Algorithm 8: IndexConstruction.QHTI(DBgraph $\mathcal{G}_{\mathcal{D}}$ )

---

```

output : T:  $\mathcal{G}_{\mathcal{D}}$  decomposed and hashed into a tree T
1 initialization:  $TempTG \leftarrow \text{NULL}$ ;  $label \leftarrow \text{NULL}$ ;  $T \leftarrow \text{NULL}$ ;  $\ell \leftarrow \text{MaxQuerySize}$ ;
2  $T.\text{Level}(1) \leftarrow \mathcal{G}_{\mathcal{D}}$ ;
3 for  $k \leftarrow 2$  to  $\ell$  do
4    $TempHG \leftarrow \text{NULL}$ ;
5   for  $h \leftarrow 1$  to  $\ell$  do
6     for  $m \leftarrow 1$  to  $|T.\text{Level}(k-1)|$  do
7        $label \leftarrow \text{Merge}(\mathcal{G}_{\mathcal{D}}[h], T.\text{Level}(k-1)[m])$ ;
8        $TempHG.\text{Add}(label)$ ;
9     end
10  end
11   $T.\text{Add}(TempHG, k)$ ;
12   $T.\text{Level}(k).\text{ApplyLinearHashing}()$ ;
13 end

```

---

For instance, given a query with a hash value of (01), only hash entries with value (01) need to be traversed. In Algorithm 9 we first generate a hash entry from  $\mathcal{G}_{\mathcal{Q}}$  (lines 2 to 4). Then, the hash value of this entry is used to fetch all entries with the same value in T (line 5). Finally, the hash key of the query is matched against the hash keys of T using the hash search operation  $\text{Get}(\cdot)$  in line 6. This  $\text{Get}(\cdot)$  sub-procedure (defined in Algorithm 10) extracts the actual entries of the hash values. Figure 5.4 illustrates an example of structuring and matching a query against the entries of  $T.\text{level}(2)$ .

### 5.2.3 Delete-QHTI

In order to delete an object  $o$  from  $\mathcal{G}_{\mathcal{D}}$ , each level of T must be traversed and each entry containing  $o$  needs to be deleted (Algorithm 11). In each level we first store all tuples that contain  $o$  (lines 2 and 3) and for each of these tuples we generate the key and delete the corresponding entry (lines 4 to 8).

### 5.2.4 Discussion

In this section, we have proposed QHTI that hashes and indexes all possible object pairs combined with their relations into buckets of a hash table. Nevertheless, QHTI does not

---

**Algorithm 9:** Search\_QHTI(QHTI-tree T, Query  $\mathcal{G}_Q$ )
 

---

**output:**  $\mathcal{M}$ : set of matches that satisfy  $\mathcal{G}_Q$

```

1 initialization:  $\mathcal{M} \leftarrow \text{NULL}$ ;  $QHash \leftarrow \text{NULL}$ ;  $HashList \leftarrow \text{NULL}$ ;  $\ell \leftarrow |\mathcal{G}_Q|$ ;
2 foreach pair  $p = (o_i, o_j) \in \mathcal{G}_Q$  with  $o_i \neq o_j$  do
3   |  $QHash.Add(\text{Merge}(p, \mathcal{G}_Q))$ ;
4 end
5  $HashList \leftarrow T.(\ell).GetHashKeys(\text{GetHash}(QHash))$ ;
6  $\mathcal{M} \leftarrow \text{Get}(HashList, QHash)$ ;
7 return  $\mathcal{M}$ ;
    
```

---



---

**Algorithm 10:** Get(Hash List  $HList$ , Hash Key  $QHash$ )
 

---

**output:**  $\mathcal{M}$ : set of hash values

```

1 if  $|HList| < 1$  then
2   | return  $\text{NULL}$ ;
3 else if  $|HList| == 1 \wedge HList == QHash$  then
4   | return  $HList.Ptr$ ;
5 else
6   |  $\mathcal{M} \leftarrow \text{NULL}$ ;
7   | for  $z \leftarrow 0$  to  $|HList|$  do
8     |   if  $HList[z] == QHash$  then
9       |   |  $\mathcal{M}.Add(HList[z].ptr)$ ;
10    |   | return  $\mathcal{M}$ ;
    
```

---

scale in terms of space as it requires exponential number of sets with the levels of T of QHTI. In general, QHTI generates many recurring sets of hash values in each level of tree T of QHTI. Accordingly, there is a great opportunity to develop an approach that extends QHTI by handling these recurring sets and represent them as a unique set which will arguably decrease the total size of T.

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

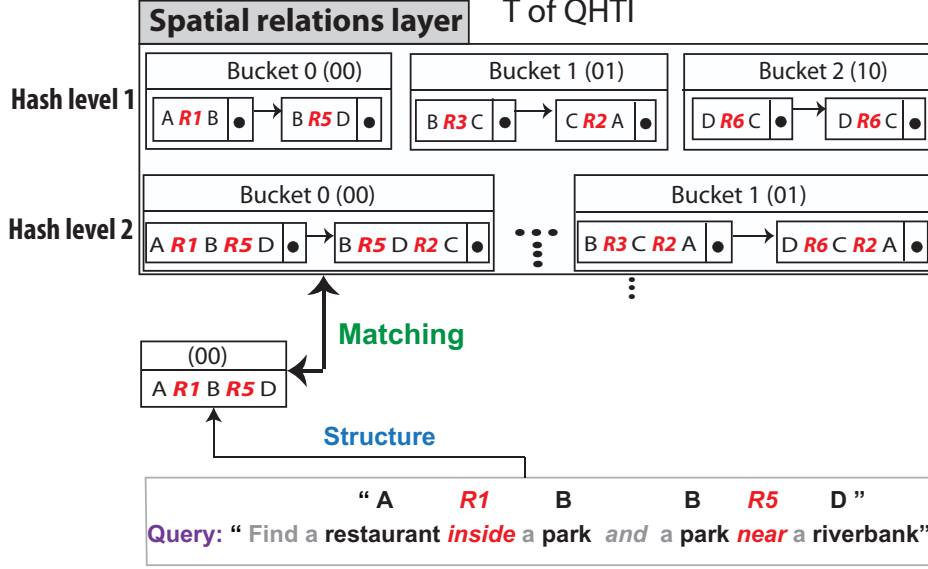


Figure 5.4: Example of structuring and matching a query against the second level of T.

### 5.3 Qualitative Hash Table Compressing

In Section 5.2.4, we pointed out that space demands of QHTI are high. Additionally, we observed that QHTI may generate many recurring sets in each level of tree (T) of QHTI. In this section, we thusly propose Qualitative Hash Table Compression (QHTC) that reduces space consumption by extracting recurring sets. QHTC summarizes these recurring sets and represents them uniquely in a hash table ( $T^C$ ). Similar to QHTI, QHTC provides the three methods: (1) Index Construction, (2) Search, and (3) Delete.

#### 5.3.1 Index Construction-QHTC

The index construction of QHTC represents all entries of T derived by QHTI as a Unique Set of Values ( $\mathcal{USV}$ ) by eliminating all recurrent sets (Figure 5.2(e)). Then, in each level of T derived by QHTI, a  $\mathcal{USV}$  is stored in the corresponding level of  $T^C$ . Moreover, QHTC computes the MBR for each entry of the  $\mathcal{USV}$ <sup>1</sup>. We note that the MBR is very important to give an approximation for sets to match, even before retrieving the actual geometries of these recurrent sets. In Figure 5.2(e) we depict the compression of the sets of QHTI in a semantic layer. We deploy linear hashing in QHTC to map each unique set into the corresponding bucket in  $T^C$ . In addition, each hash entry is assigned three additional

<sup>1</sup>We calculate the MBR by accumulating over the centroids of each recurrent set.



---

**Algorithm 11:** Delete\_QHTI(QHTI\_tree T, Object  $o$ )
 

---

**output:** T': T without any occurrence of  $o$

```

1 initialization:  $TMP \leftarrow \text{NULL}$ ;
2 for  $k \leftarrow 1$  to T.MaxLevel do
3    $TMP \leftarrow \text{T.Level}(k).\text{GetPairs}(o)$ ;
4   foreach  $tmp \in TMP$  do
5     if  $tmp.Id == o.Id$  then
6        $\text{T.Level}(k).\text{LinearHash.Delete}(\text{GetHash}(tmp), tmp)$ ;
7     end
8   end
9 end
    
```

---

variables: 1)  $NR$ : the number of recurrences, 2)  $MBR$ : the minimum bounding rectangle of the recurring sets, and 3)  $Id$ : a unique id to the corresponding geometries in the database  $\mathcal{D}$ . As introduced before (cf. Section 4.3), we denoted the geometries part of  $\mathcal{D}$  by  $F_{\mathcal{D}}$ .

Algorithm 12 details the steps to construct the index. Given a hash table T generated by QHTI we hash and map all recurrent sets in each level  $k$  into so-called buckets (lines 2 to 4). When a recurrent set with the same hash key and value already exists in  $T^C$  (lines 5 to 7), we increment the counter  $NR$  for this hash entry (line 8). Subsequently, the pointer to the actual geometries of the objects in  $F_{\mathcal{D}}$  is added to a database table  $DBh$  that maintains all pointers corresponding to  $F_{\mathcal{D}}$  (line 9). Additionally, the  $MBR$  is updated to include the new geometry of the matched entry (line 10). Otherwise, a recurrent set is hashed and mapped to the corresponding bucket, its  $NR$  value is initialized with 1 and a pointer is added to  $DBh$ , its  $Id$  is initialized by a new unique identifier, and its  $MBR$  is updated (lines 12 to 17). Finally,  $B^+$ -trees are applied on the pointers ( $Id$ 's) of  $DBh$  to allow retrieving the corresponding geometries in logarithmic time (line 20).

A detailed example of constructing the index of  $T^C.\text{Level}(1)$  is depicted in Figure 5.5. Figure 5.5(a) shows two entries in  $T.\text{Level}(1)$  (marked with green color) that form a recurring set and Figure 5.5(b) shows how they are represented in  $T^C.\text{Level}(1)$ .

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---



---

### Algorithm 12: IndexConstruction.QHTC(QHTI\_tree T, GeomDB $F_{\mathcal{D}}$ )

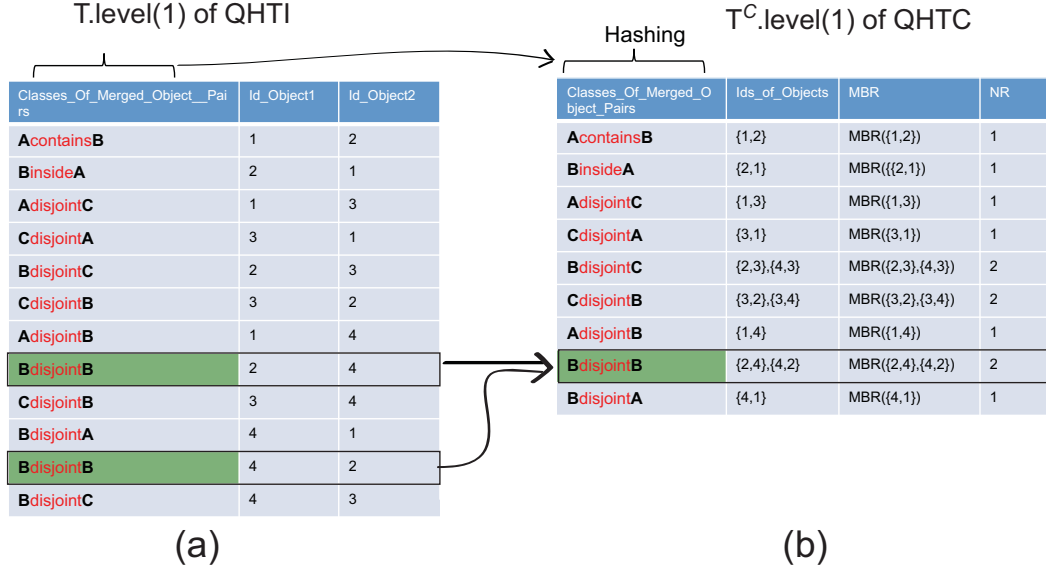
---

```

output:  $T^C$ : all unique hash entries in T,
            $DBh$ : according pointers  $F_{\mathcal{D}}$  for entries  $T^C$ 
1 initialization:  $T^C \leftarrow \text{NULL}$ ;  $HashList \leftarrow \text{NULL}$ ;  $SPairs \leftarrow \text{NULL}$ ;  $DBh \leftarrow \text{NULL}$ ;
    $c \leftarrow 0$ ;  $h \leftarrow \text{NULL}$ ;
2 for  $k \leftarrow 1$  to T.MaxLevel do
3    $SPairs \leftarrow T[k]$ ;
4   foreach  $SP \in SPairs$  do
5      $h \leftarrow \text{GetHash}(SP)$ ;
6      $HashList \leftarrow T^C.\text{Level}(k).\text{GetHashKeys}(h)$ ;
7     if  $\text{Get}(HashList, SP) \neq \text{NULL}$  then
8        $T^C.\text{Level}(k).SP.NR++$ ;
9        $DBh.\text{Add}(SP.Id, F_{\mathcal{D}}.\text{Geometries})$ ;
10       $T^C.\text{Level}(k).SP.\text{Update}(\text{MBR}) \leftarrow DBh.\text{GetMBR}(SP.Id, F_{\mathcal{D}})$ ;
11    else
12       $c++$ ;
13       $T^C.\text{Level}(k).\text{Hash\&Insert}(SP)$ ;
14       $T^C.\text{Level}(k).SP.NR \leftarrow 1$ ;
15       $T^C.\text{Level}(k).SP.Id \leftarrow c$ ;
16       $DBh.\text{Level}(k).\text{Add}(SP.Id, F_{\mathcal{D}}.\text{Geometries})$ ;
17       $T^C.\text{Level}(k).SP.\text{Update}(\text{MBR}) \leftarrow DBh.\text{GetMBR}(SP.Id, F_{\mathcal{D}})$ ;
18    end
19  end
20   $DBh.\text{Level}(k).Ids.\text{Apply.B}^+\text{-tree}()$ ;
21 end

```

---



**Figure 5.5:** An example: the index construction of the first level of  $T^C$ .

### 5.3.2 Search-QHTC

QHTC only requires visiting  $T^C$  entries that have the same hash value as  $\mathcal{G}_Q$  in order to match a spatial query  $\mathcal{G}_Q$ .

Algorithm 13 outlines the steps for matching a user query against the entries of  $T^C$ . Firstly,  $\mathcal{G}_Q$  is restructured as a hash entry ( $QHash$ ) (lines 2 to 4). The hash value is then derived from  $QHash$  and used to retrieve entries of  $T^C$  which have the same hash value. These are stored in the linked-list ( $HashList$ ) (line 5). Secondly,  $HashList$  and the key of  $QHash$  are passed to the  $Get()$  sub-procedure that finds the entry of  $T^C$  with the same key (line 6). Unlike QHTI, in QHTC the  $Get()$  sub-procedure returns a single entry at most, due to the fact that the entries of  $T^C$  are unique. If a query is matched, the corresponding MBR and  $NR$  are first retrieved from the matched entry of  $T^C$  (lines 7 and 8). Lastly, the actual geometries of the matched entries are retrieved from  $F_D$  via  $DBh$  (line 9).

### 5.3.3 Delete-QHTC

Deleting an object  $o$  from  $T^C$  requires traversing each level of  $T^C$  and removing entries containing  $o$  (Algorithm 14). QHTC iterates over each level of  $T^C$  and extracts all tuples containing  $o$  (lines 2 and 3). Now, for each extracted entry, the matches are retrieved

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---



---

**Algorithm 13:** Search\_QHTC(QHTC\_tree  $T^C$ , Query  $\mathcal{G}_Q$ , DBPointers  $DBh$ , GeomDB  $F_D$ )

---

**output:**  $\mathcal{M}$ : a set of matches that satisfy  $\mathcal{G}_Q$

```

1 initialization:  $\mathcal{M} \leftarrow \text{NULL}$ ;  $level \leftarrow |\mathcal{G}_Q|$ ;  $QHash \leftarrow \text{NULL}$ ;  $MatchEnt \leftarrow \text{NULL}$ ;
    $HashList \leftarrow \text{NULL}$ ;
2 foreach  $Pair\ p \in \mathcal{G}_Q$  do
3   |  $QHash.Add(\text{MergePairAndRelation}(p))$ ;
4 end
5  $HashList \leftarrow T^C(level).GetHashKeys(\text{GetHash}(QHash))$ ;
6  $MatchEnt \leftarrow \text{Get}(HashList, QHash)$ ;
7 if  $MatchEnt \neq \text{NULL}$  then
8   |  $MatchEnt.Returns(NR, \text{MBR})$ ;
9   |  $\mathcal{M}.Add(DBh.GetGeometries(MatchEnt.Id, F_D))$ ;
10  | return  $\mathcal{M}$ ;
11 end
```

---

and stored in  $\mathcal{M}$  (lines 4 and 5). Next, QHTC goes through each entry ( $mat$ ) of  $\mathcal{M}$  to find the entries that have the same  $Id$  as  $o.Id$  (lines 6 and 7). When  $mat.Id$  equals  $o.Id$  and its number of recurrences  $NR > 1$ , then  $NR$  is decremented, the  $MBR$  value of  $mat$  is updated, and its pointers of  $DBh$  and  $B^+$ -trees are deleted (lines 8 to 12). Otherwise, the  $mat$  entry and its pointers are completely eliminated from  $T^C$  (lines 13 to 17).

### 5.3.4 Discussion

As an extension of QHTI, we have proposed QHTC approach that has been aimed to reduce the vast number of (recurring) entries in tree ( $T$ ) generated by QHTI. Although QHTC may have a much smaller number of entries, each QHTC entry contains pointers to a list of object pairs to the hash value of the entry. As a result, the number of object pairs that QHTC has to keep track of is exact the same as that of QHTI.

However, the main goal of QHTC was to accelerate the matching of QSQs on a qualitative level but not a quantitative one. This procedure gave QHTC an ability to retrieve the  $MBR$  and the number of a query matches even before retrieving the actual geometries for these matches. Moreover, QHTC keeps track of the corresponding object pairs by using a list of numeric  $Id$ 's that considerably require less space than storing

---

**Algorithm 14:** Delete\_QHTC(QHTC\_tree  $T^C$ , DBPointers  $DBh$ , Object  $o$ , GeomDB  $F_{\mathcal{D}}$ )

---

```

output:  $T^C$  :  $T^C$  without  $o$ 
1 initialization:  $\mathcal{M} \leftarrow \text{NULL}$ ;  $TMP \leftarrow \text{NULL}$ ;
2 for  $l \leftarrow 1$  to  $T^C.\text{MaxLevel}$  do
3    $TMP \leftarrow T^C.\text{Level}(l).\text{GetPairs}(o)$ ;
4   foreach  $tmp \in TMP$  do
5      $\mathcal{M} \leftarrow \text{Search\_QHTC}(TMP, tmp, DBh, F_{\mathcal{D}})$ ;
6     foreach  $mat \in \mathcal{M}$  do
7       if  $mat.Id == o.Id$  then
8         if  $(mat.NR) > 1$  then
9            $T^C.\text{Level}(l).mat.NR \leftarrow NR-1$ ;
10           $T^C.\text{Level}(l).SP.\text{Update}(\text{MBR}) \leftarrow DBh.\text{GetMBR}(mat, F_{\mathcal{D}})$ ;
11           $DBh.\text{Level}(l).B^+\text{-tree.Delete}(mat.Id)$ ;
12           $DBh.\text{Level}(l).\text{Delete}(o.Id, F_{\mathcal{D}}.\text{Geometries})$ ;
13        else
14           $T^C.\text{Level}(l).\text{LinearHash.Delete}(mat)$ ;
15           $DBh.\text{Level}(l).B^+\text{-tree.Delete}(mat.Id)$ ;
16           $DBh.\text{Level}(l).\text{Delete}(o.Id, F_{\mathcal{D}}.\text{Geometries})$ ;
17        end
18      end
19    end
20  end
21 end
    
```

---

the labels of object pairs and their relations themselves.

Nevertheless, the efficiency of QHTC depends on the amounts of the recurrent entries of  $T$ . Consequently, QHTC becomes impractical approach when the  $T$  entries do not possess many recurrences. Alternatively, the recurrences of the labels of the relations or object pairs can be determined in order to increase the possibility of finding recurrences (or repeated sets). Therefore, variants of QHTC that represent either the labels of relations or object pairs as unique sets can be developed. The development of such variants will lead to decrease the amounts of the recurring labels of relations or object pairs in  $T$  respectively.

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---

### 5.4 Qualitative Hash Table Compressing of Qualitative Models

In the previous section, we have developed QHTC as an extension to QHTI. Furthermore, we have indicated that determining either labels of relations or object pairs instead of combining both, might increase the possibility of finding recurrences (cf. Section 5.3.4). Especially, qualitative models and object pairs usually have a limited number of labels of relations or object pairs respectively. For instance, the 9-intersection model contains eight spatial relations and thus eight labels which implies a higher possibility of finding recurrences. In this section, we only focus on determining the recurring sets of labels of relations and we thusly propose Qualitative Hash Table Compressing of Qualitative Models (QHTC<sup>M</sup>) as a variant of the original QHTC. QHTC<sup>M</sup> provides three operations: Indexing Construction-QHTC<sup>M</sup>, Search-QHTC<sup>M</sup>, and Delete-QHTC<sup>M</sup>.

#### 5.4.1 Index Construction-QHTC<sup>M</sup>

The key idea behind the Index Construction (IC) of QHTC<sup>M</sup> is to aggregate the recurrent sets of labels, i.e., relations. These labels can be used to index the underlying object tuples labels. QHTC<sup>M</sup> stores unique relations in the hash table ( $T^M.SpRel$ ), where each entry points to the corresponding object tuples stored in another database table ( $T^M.OPairs$ ). In particular, linear hashing is deployed in QHTC<sup>M</sup> to map each unique set of relations into the corresponding bucket in  $T^M.SpRel$ . In addition, each hash entry has two additional variables: 1) *NR*: the number of recurrences and 2) *Id*: a unique id to the corresponding object tuples in the database  $\mathcal{D}$ .

Algorithm 15 illustrates construction of the index of QHTC<sup>M</sup>. The Algorithm runs through each level of  $T$  of QHTI, extracts spatial relations and object tuples, and stores them in two array variables *SpatialRels* and *Pairs* (lines 2 to 4). Next, hash values are derived from *SpatialRels* entries that are used to map *SpatialRels* entries (keys) into corresponding buckets (lines 5 to 8). However, when an entry does already exist in the hash table, then its *NR* value is incremented and a pointer is added to its associated object tuples (lines 9 to 11). Otherwise, an entry is inserted into the hash table (line 13), its *NR* value is initialized by 1 (line 14), its *Id* is initialized by a new unique identifier (lines 15 and 16), and eventually a pointer is added to its associated object tuples (line 17).

---

**Algorithm 15:** IndexConstruction\_QHTC<sup>M</sup>(QHTI\_tree T)

---

**output:**  $T^M$ : aggregated the relations of  $T$  pointing to their object tuples

```

1 initialization:  $T^M \leftarrow \text{NULL}$ ;  $HashList \leftarrow \text{NULL}$ ;  $SpatialRels \leftarrow \text{NULL}$ ;
    $Pairs \leftarrow \text{NULL}$ ;  $c \leftarrow 0$ ;  $OP \leftarrow \text{NULL}$ ;  $SR \leftarrow \text{NULL}$ ;
2 for  $k \leftarrow 1$  to  $T.\text{MaxLevel}$  do
3    $SpatialRels \leftarrow T[k].\text{GetRelations}()$ ;
4    $Pairs \leftarrow T[k].\text{GetObjPairs}()$ ;
5   for  $u \leftarrow 1$  to  $|SpatialRels|$  do
6      $SR \leftarrow SpatialRels[u]$ ;
7      $OP \leftarrow Pairs[u]$ ;
8      $HashList \leftarrow T^M.\text{Level}(k).SpRel.\text{GetHashKeys}(\text{GetHash}(SR))$ ;
9     if  $\text{Get}(HashList, SR) \neq \text{NULL}$  then
10       $T^M.\text{Level}(k).SpRel.SR.NR++$ ;
11       $T^M.\text{Level}(k).OPairs.\text{Add}(SR.Id, OP)$ ;
12    else
13       $T^M.\text{Level}(k).SpRel.\text{Hash\&Insert}(SR)$ ;
14       $T^M.\text{Level}(k).SpRel.SR.NR \leftarrow 1$ ;
15       $c++$ ;
16       $T^M.\text{Level}(k).SpRel.SR.Id \leftarrow c$ ;
17       $T^M.\text{Level}(k).OPairs.\text{Add}(SR.Id, OP)$ ;
18    end
19  end
20 end

```

---

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---

### 5.4.2 Search-QHTC<sup>M</sup>

QHTC<sup>M</sup> provides a hierarchical matching procedure which consists of two phases. In the first phase, QHTC<sup>M</sup> attempts to match the relations of a spatial query  $\mathcal{G}_Q$  against the relations of  $T^M$ . Once a matching is found in the hash table, the second phase of QHTC<sup>M</sup> commences by using the relations to retrieve all its related object tuples. Finally, the retrieved object tuples of  $T^M.OPairs$  are matched by the object tuples of  $\mathcal{G}_Q$ .

Algorithm 16 details the steps for matching  $\mathcal{G}_Q$  against the entries of  $T^M$ . It first stores the relations and object tuples of  $\mathcal{G}_Q$  in two variables  $SR$  and  $OP$  respectively (lines 2 and 3). Afterwards, the hash value of  $SR$  is generated and used to fetch the hash keys of  $T^M.SpRel$  with the same hash value (line 4). Subsequently, the hash keys are matched by  $SR$  through the  $Get()$  sub-procedure (line 5). If  $SR$  is found, then the corresponding object tuples and their  $g^{Id's}$  (Id's pointing to geometries) are retrieved (lines 6 to 8).

---

**Algorithm 16:** Search\_QHTC<sup>M</sup>(QHTC<sup>M</sup>\_tree  $T^M$ , Query  $\mathcal{G}_Q$ )

---

**output:**  $\mathcal{M}$ : a set of matches that satisfy  $\mathcal{G}_Q$

---

```

1 initialization:  $\mathcal{M} \leftarrow \text{NULL}$ ;  $level \leftarrow |\mathcal{G}_Q|$ ;  $SR \leftarrow \text{NULL}$ ;  $MatchEntry \leftarrow \text{NULL}$ ;
    $OP \leftarrow \text{NULL}$ ;  $HashList \leftarrow \text{NULL}$ ;
2  $SR \leftarrow \mathcal{G}_Q.GetRelations()$ ;
3  $OP \leftarrow \mathcal{G}_Q.GetObjPairs()$ ;
4  $HashList \leftarrow T^M(level).SpRel.GetHashKeys(GetHash(SR))$ ;
5  $MatchEntry \leftarrow Get(HashList, SR)$ ;
6 if  $MatchEntry \neq \text{NULL}$  then
7   | return  $\mathcal{M} \leftarrow T^M.OPairs.Level(level).Search(MatchEntry.Id, OP)$ ;
8 end
```

---

### 5.4.3 Delete-QHTC<sup>M</sup>

To delete an object  $o$  from  $T^M$ , all levels of  $T^M$  need to be traversed. Meaning all entries that match the label and  $Id$  of  $o$  are deleted within each level of  $T^M$ . Algorithm 17, outlines the steps to delete  $o$  from  $T^M$ . Firstly, the relations of  $T^M$  are extracted for each level of the tree, and stored in  $SR$  (lines 2 and 3). Secondly, we go through each entry  $sr$  of  $SR$ , use its id to fetch related object tuples, and eventually store them in  $MatchedPairs$  (lines 4 and 5). We similarly run through each entry  $mat$  of



## 5.4 Qualitative Hash Table Compressing of Qualitative Models

---

**Algorithm 17:** Delete\_QHTC<sup>M</sup>(QHTC<sup>M</sup>\_tree T<sup>M</sup>, Object *o*)

---

```

output: TM: TM without o
1 initialization: mat ← NULL; SR ← NULL; MatchedPairs ← NULL;
2 for l ← 1 to TM.MaxLevel do
3   SR ← TM.Level(l).SpRe.GetRelations();
4   foreach sr ∈ SR do
5     MatchedPairs ← TM.OPairs.Level(l).GetObjPairs(sr.Id);
6     foreach mat ∈ MatchedPairs do
7       if mat.Id == o.Id then
8         if (mat.NR) ≤ 1 then
9           TM.Level(l).SpRel.LinearHash.Delete(sr);
10          TM.Level(l).OPairs.Delete(mat.Id);
11         else
12           TM.Level(l).SpRel.sr.NR ← NR - 1;
13           TM.Level(l).OPairs.Delete(mat.Id);
14         end
15       end
16     end
17   end
18 end

```

---

*MatchedPairs* (line 6) to erase any entry with the same *Id* of *o* (line 7). When only one *MatchedPairs* entry is detected, then the *sr* and *mat* entries are completely deleted from T<sup>M</sup> (lines 8 to 10). Otherwise, the *NR* of *sr* is decremented and the *mat* entry is removed from T<sup>M</sup> (lines 11 to 14).

### 5.4.4 Discussion

In this section, we have described QHTC<sup>M</sup> as an extension to QHTC. We have particularly pointed out that QHTC<sup>M</sup> determines labels of relations instead of the combined labels of relations and object pairs as it is done by QHTC. This procedure might boost the possibility of discovering recurrence sets. Nevertheless, QHTC<sup>M</sup> has a limitation as it allows for only reduction of qualitative dimensions, whereas the labels of object pairs are completely stored in the database. Moreover, QHTC<sup>M</sup> is based on a two-stage index

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---

structure which means that the implementation of such an index structure is much more complex than the index structure induced by QHTC.

### 5.5 Qualitative Hash Table Compressing of Object Pairs

Quite similar to  $\text{QHTC}^M$  and with the same foundations and motivation, we propose a Qualitative Hash Table Compressing of Object Pairs ( $\text{QHTC}^P$ ) as a variant of QHTC.  $\text{QHTC}^P$  is a two-levels index structure, in which the recurrent sets of the labels of object pairs tuples are aggregated and then used to index the related labels of relations.  $\text{QHTC}^P$  offers three operations: Index Construction- $\text{QHTC}^P$ , Search- $\text{QHTC}^P$ , and Delete- $\text{QHTC}^P$ .

#### 5.5.1 Index Construction- $\text{QHTC}^P$

The Index Construction (IC) of  $\text{QHTC}^P$  is built based on  $T$ , the output of QHTI. Very similar to  $\text{QHTC}^M$  the IC of  $\text{QHTC}^P$  stores object pairs and the relations in each level of tree. Generally speaking, IC aggregates the repeated sets of the labels of object pairs which are used to index their relations labels. The IC steps of  $\text{QHTC}^P$  are detailed in Algorithm 18. The algorithm iterates each level of  $T$ , extracts spatial relations and object pairs, and stores them in two array variables *SpatialRels* and *Pairs* respectively (lines 2 to 4). Afterwards, hash value is computed from each entry of *Pairs* and then mapped into a related bucket in the hash table  $T^P.OPairs$  (lines 5 to 8). If only the entry exists with same hash value and key, then its *NR* value is incremented and its *Id's* and  $g^{Id's}$  (Id's pointing to geometries) are added into spatial relations database table ( $T^P.SpRel$ ) (lines 9 to 12). Otherwise, the entry is inserted into the hash table (line 15), its *NR* value is initialized by 1 (line 16), its *Id* is initialized by a new unique identifier (line 17), and eventually pointers are added to its associated spatial relations tuples in  $T^P.SpRel$  (lines 18 and 19).

#### 5.5.2 Search- $\text{QHTC}^P$

A two-stage hierarchal matching is performed by  $\text{QHTC}^P$  to answer a spatial query  $\mathcal{G}_Q$  (Algorithm 19). Initially, the level of  $T^P$  is determined by the size of  $\mathcal{G}_Q$  and then the object pairs and the relations are obtained from  $\mathcal{G}_Q$  (lines 1 to 3). Subsequently, the labels of object pairs of  $\mathcal{G}_Q$  are matched against the entries of the hash table  $T^P.OPairs$  (lines 4 and 5). In case a match exists (line 6), then its *Id* and the relations of  $\mathcal{G}_Q$  are

---

**Algorithm 18:** IndexConstruction\_QHTC<sup>P</sup>(QHTI\_tree T)

---

**output:**  $T^P$  : Aggregated object pairs of  $T$  pointing to their relations

```

1 initialization:  $T^P \leftarrow \text{NULL}$ ;  $SpatialRels \leftarrow \text{NULL}$ ;  $Pairs \leftarrow \text{NULL}$ ;  $c \leftarrow 0$ ;
    $PO \leftarrow \text{NULL}$ ;  $SR \leftarrow \text{NULL}$ ;  $HashList \leftarrow \text{NULL}$ ;
2 for  $k \leftarrow 1$  to  $T.\text{MaxLevel}$  do
3    $SpatialRels \leftarrow T[k].\text{GetRelations}()$ ;
4    $Pairs \leftarrow T[k].\text{GetObjPairs}()$ ;
5   for  $u \leftarrow 1$  to  $Pairs.Length$  do
6      $SR \leftarrow SpatialRels[u]$ ;
7      $PO \leftarrow Pairs[u]$ ;
8      $HashList \leftarrow T^P.\text{Level}(k).OPairs.\text{GetHashKeys}(\text{GetHash}(PO))$ ;
9     if  $\text{Get}(HashList, PO) \neq \text{NULL}$  then
10       $T^P.\text{Level}(k).OPairs.PO.NR++$ ;
11       $T^P.\text{Level}(k).SpRel.Add(PO.Id, SR)$ ;
12       $T^P.\text{Level}(k).SpRel.Add(PO.Geoms.Id, SR)$ ;
13    else
14       $c++$ ;
15       $T^P.\text{Level}(k).OPairs.\text{Hash\&Insert}(OP)$ ;
16       $T^P.\text{Level}(k).OPairs.OP.NR \leftarrow 1$ ;
17       $T^P.\text{Level}(k).OPairs.OP.Id \leftarrow c$ ;
18       $T^P.\text{Level}(k).SpRel.Add(PO.Id, SR)$ ;
19       $T^P.\text{Level}(k).SpRel.Add(PO.Geoms.Id, SR)$ ;
20    end
21  end
22 end

```

---

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---

matched against the  $Id$ 's and the relations of  $T^P.SpRel$  entries (line 7). Finally, the matches and their geometric  $Id$ 's are retrieved (line 7).

---

**Algorithm 19:** Search\_QHTC<sup>P</sup>(QHTC<sup>P</sup>\_tree  $T^P$ , Query  $\mathcal{G}_Q$ )

---

**output:**  $\mathcal{M}$ : a set of matches that satisfy  $\mathcal{G}_Q$

```

1 initialization:  $\mathcal{M} \leftarrow \text{NULL}$ ;  $level \leftarrow |\mathcal{G}_Q|$ ;  $SR \leftarrow \text{NULL}$ ;  $PO \leftarrow \text{NULL}$ ;
    $HashList \leftarrow \text{NULL}$ ;  $MatchEntry \leftarrow \text{NULL}$ ;
2  $SR \leftarrow \mathcal{G}_Q.GetRelations()$ ;
3  $OP \leftarrow \mathcal{G}_Q.GetObjPairs()$ ;
4  $HashList \leftarrow T^P(Level).OPairs.GetHashKeys(GetHash(OP))$ ;
5  $MatchEntry \leftarrow Get(HashList, OP)$ ;
6 if  $MatchEntry \neq \text{NULL}$  then
7   | return  $\mathcal{M} \leftarrow T^P.SpRel.Level(level).Search(MatchEntry.Id, SR)$ ;
8 end
```

---

### 5.5.3 Delete-QHTC<sup>P</sup>

In QHTC<sup>P</sup>, deleting an object  $o$  from  $T^P$  is done by removing the entries which have the same  $Id$  and label of  $o$  in each level of  $T^P$  (Algorithm 20). For each level of  $T^P$ , QHTC<sup>P</sup> starts by finding object tuples that contain the object label of  $o$  (lines 2 and 3). Next, it iterates over each entry of the found object tuples and deletes the entries containing  $o.Id$  (lines 4 to 18).

---

**Algorithm 20:** Delete- $\text{QHTC}^P(\text{QHTC}^P\_tree\ T^P, \text{Object } o)$ 


---

```

output:  $\text{QHTC}^P$ :  $\text{QHTC}^P$  without  $o$ 
1 initialization:  $mat \leftarrow \text{NULL}$ ;  $PO \leftarrow \text{NULL}$ ;  $MatchedRels \leftarrow \text{NULL}$ ;
2 for  $l \leftarrow 1$  to  $T^P.\text{MaxLevel}$  do
3    $PO \leftarrow T^P.\text{Level}(l).OPairs.\text{GetObjPairs}(o)$ ;
4   foreach  $po \in PO$  do
5      $MatchedRels \leftarrow T^P.SpRe.\text{Level}(l).\text{GetRelations}(po.Id)$ ;
6     foreach  $mat \in MatchedRels$  do
7       if  $mat.Geoms.Id == o.Id$  then
8         if  $(mat.NR) \leq 1$  then
9            $T^P.\text{Level}(l).SpRel.\text{LinearHash}.\text{Delete}(mat.Id)$ ;
10           $T^P.\text{Level}(l).OPairs.\text{Delete}(po)$ ;
11         else
12            $T^P.\text{Level}(l).OPairs.sr.NR \leftarrow NR-1$ ;
13            $T^P.\text{Level}(l).SpRel.\text{Delete}(mat.Id)$ ;
14         end
15       end
16     end
17   end
18 end

```

---

## 5.6 Summary

In this chapter we have presented five optimization indexing approaches to enhance the space and time scalability of the spatial query processing. The properties and capabilities vary from one approach to the next. We therefore summarize in Table 5.1 these properties and capabilities for the proposed approaches. Table 5.1 conveys that all the approaches are dynamic, in the sense that they allow for updating (e.g., delete) the database. In particular, HITBT has an ability and flexibility to answer the queries that have variant number of binary relations. For example, a query may contain a single relation (e.g., *disjoint*) between one object pair and two relations (e.g., *inside* and *north*) between another object pair. In fact such approach might be suitable and applicable to deal with the queries that come in the form of verbal descriptions.

## 5. OPTIMIZING INDEXING APPROACHES FOR SPATIAL DATABASES

---

**Table 5.1:** Comparison between the five indexing approaches.

	Dynamic	Relations Flexible	Dimensions Reduc.	Provide Reduc.	MBR Approx.	Indexing
HITBT	yes	yes	no	no	no	B <sup>+</sup> -trees
QHTI	yes	no	yes	no	no	Linear Hashing
QHTC	yes	no	yes	yes	yes	Linear Hashing and B <sup>+</sup> -trees
QHTC <sup>M</sup>	yes	no	partially	partially	no	Linear Hashing and B <sup>+</sup> -trees
QHTC <sup>P</sup>	yes	no	partially	partially	no	Linear Hashing and B <sup>+</sup> -trees

In turn, QHTI includes dimensions reduction property. QHTC offers dimensions and qualitative data reduction as well as an MBR approximation for the queries. Finally QHTC<sup>M</sup> and QHTC<sup>P</sup> partially provide dimensions and qualitative data reductions.

QHTI, QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> are only able to answer the queries that have the same number of relations between object pairs. Hence, these methods might be useful to cope with the queries that come in the form of visual descriptions (e.g., query-by-sketch).

# Chapter 6

## Implementation and Applications

In this chapter, we present the components of our system that we call **QualEnabler** and depict in Figure 6.1. **QualEnabler** consists of six components: (1) PostGIS (see Section 6.1), (2) A Qualitative Spatial Layer (see Section 6.2), (3) Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Clustering (see Section 6.3), (4) Indexing Approaches (see Section 6.4), (5) Client-Side Interfaces (see Section 6.5), and (6) System Evaluation (see Section 6.6). Some of the system components have been implemented in Java<sup>1</sup> or PL/pgSQL<sup>2</sup> or a combination of them. We select Java as it is open source and platform independent. Furthermore, it offers several ready-to-use libraries that provide a rich set of data-structures (e.g., hash tables) and built-in functions. In turn, we select PL/pgSQL, since it supports functions and data-structures that are optimized to PostgreSQL. Finally, the client-side interfaces are developed using PHP<sup>3</sup>, XHTML<sup>4</sup>, and Java, since they are efficient application development languages.

### 6.1 PostGIS: A Spatial Layer

In order to store, manage, and query spatial data we have used PostgreSQLv9.0<sup>5</sup>, an open source and powerful Spatial Data-Base Management System (SDBMS).

<sup>1</sup>Java: <http://www.oracle.com/us/technologies/java/overview/index.html>

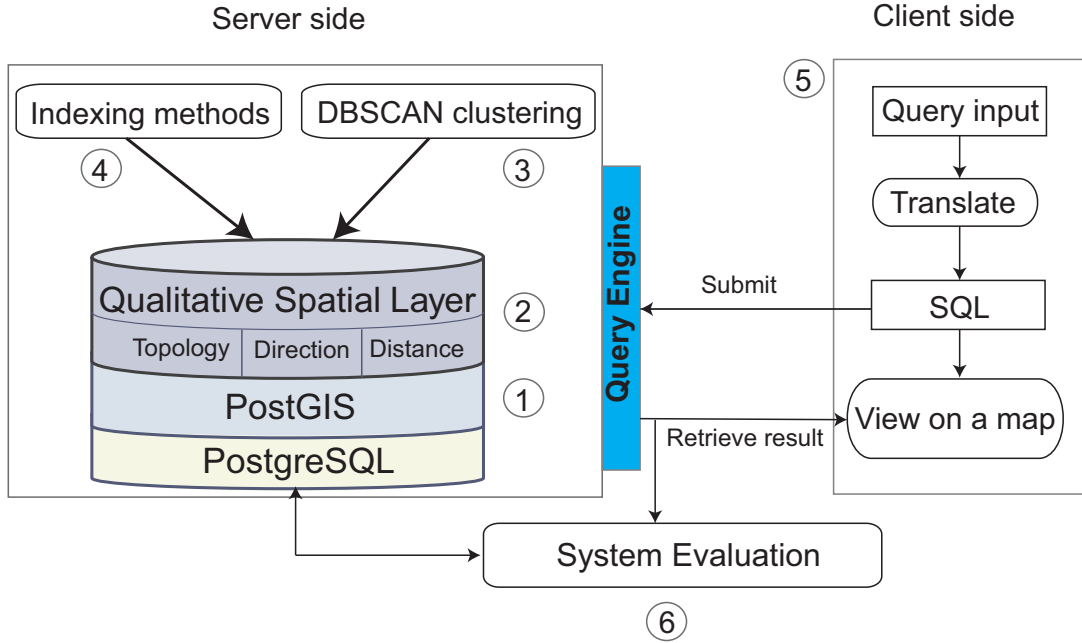
<sup>2</sup>PL/pgSQL: <http://www.postgresql.org/docs/9.0/static/plpgsql.html>

<sup>3</sup>PHP: <http://www.php.net/>

<sup>4</sup>XHTML: <http://www.w3schools.com/html/default.asp>

<sup>5</sup>PostgreSQL: <http://www.postgresql.org/ftp/source/v9.0.0/>

## 6. IMPLEMENTATION AND APPLICATIONS



**Figure 6.1:** An overview of system architecture.

PostGIS 2.0<sup>1</sup> has been installed as spatial extension of PostgreSQL to deal with the spatial data (e.g., geometries). In particular, PostGIS is developed based on the Geometry Object Model (GOM) that has been described in Section 4.3. Therefore, PostGIS offers the predicates described in Section 3.1.1 and provided by the GOM as well as many extra powerful predicates to deal with spatial data.

In Table 6.1, we outline some of these predicates with their descriptions that are essential to develop our qualitative models and approaches.

### 6.1.1 Integrating Qualitative Spatial Models into PostGIS

As we mentioned in Section 4.3, the GOM only supports the Dimensionally Extended 9-Intersection Model (DE-9IM) through the topological predicates. Thus, we integrate the directional and distance models into PostGIS to allow the qualitative usage of the directional and distance aspects. We first integrate two kinds of direction models: (1) cone-based direction and (2) Cardinal Direction Model (CDM) for extended objects. We particularly use the ST\_Azimuth function (see Table 6.1) to integrate the cone-based direction model. We call the function that abstracts the cone-based directional relations

<sup>1</sup>PostGIS: <http://postgis.net/docs/manual-2.0/>



**Table 6.1:** The predicates and their descriptions from <http://postgis.net/docs/manual-2.0/>.

Predict	Description
ST_XMax	Returns X maxima of a bounding box 2d or 3d or a geometry.
ST_YMax	Returns Y maxima of a bounding box 2d or 3d or a geometry.
ST_XMin	Returns X minima of a bounding box 2d or 3d or a geometry.
ST_YMin	Returns Y minima of a bounding box 2d or 3d or a geometry.
ST_MaxDistance	Returns the 2-dimensional largest distance between two geometries in projected units.
ST_ConcaveHull	The concave hull of a geometry represents a possibly concave geometry that encloses all geometries within the set.
ST_ConvexHull	The convex hull of a geometry represents the minimum closed geometry that encloses all geometries within the set.
ST_Azimuth	Returns the angle in radians from the horizontal of the vector defined by point A and point B. Angle is computed clockwise from down-to-up: on the clock: 12=0; 3=PI/2; 6=PI; 9=3PI/2.
ST_Envelope	Returns a valid geometry (POINT, LINESTRING or POLYGON) representing the bounding box of the geometry.

as `Dir_Cones_Abstracter()`. Afterwards, we integrate the CDC into PostGIS, based on two sets of functions: (1) the Envelope that computes the axis-aligned Minimum Bounding Rectangle (MBR) of a reference object (A) and a primary object (B) and (2) `ST_XMax`, `ST_YMax`, `ST_XMin`, and `ST_YMin` that return the two (min and max) endpoints of the MBR major diagonals. Based on the aforementioned function, a single and/or a multi-tile directional relations can be computed. For example, object B is *north east* of object A if  $B.X_{Min} > A.X_{Min}$  and  $B.Y_{Min} > A.Y_{Min}$ . We call the function that abstracts the direction relations for extended objects as `CDM_Abstracter()`. Regarding the absolute distance model, we develop four distance predicates (*ZeroDist*, *near*, *medium*, and *far*) to compute the distance relations between any two objects (e.g., A and B) in 2D space. These predicates are encapsulated in a distance function that we call `Abs_Dist_Abstracter()`.

## 6. IMPLEMENTATION AND APPLICATIONS

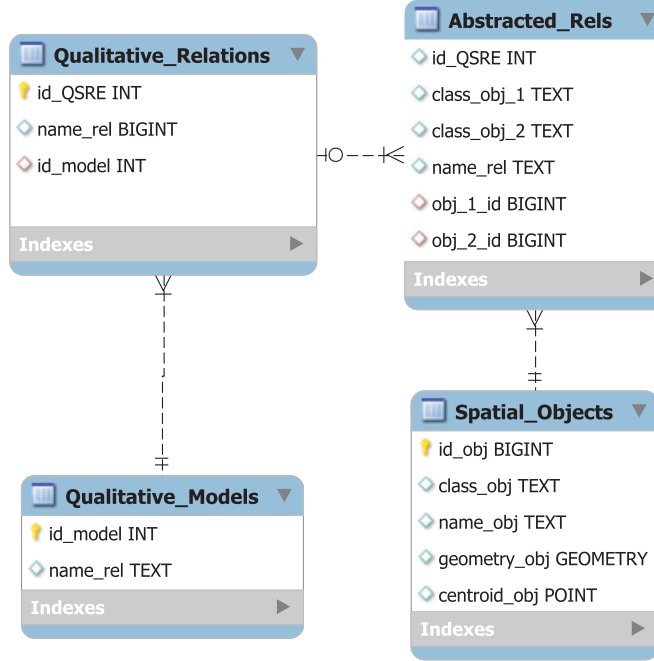


Figure 6.2: Qualitative spatial layer database schema design.

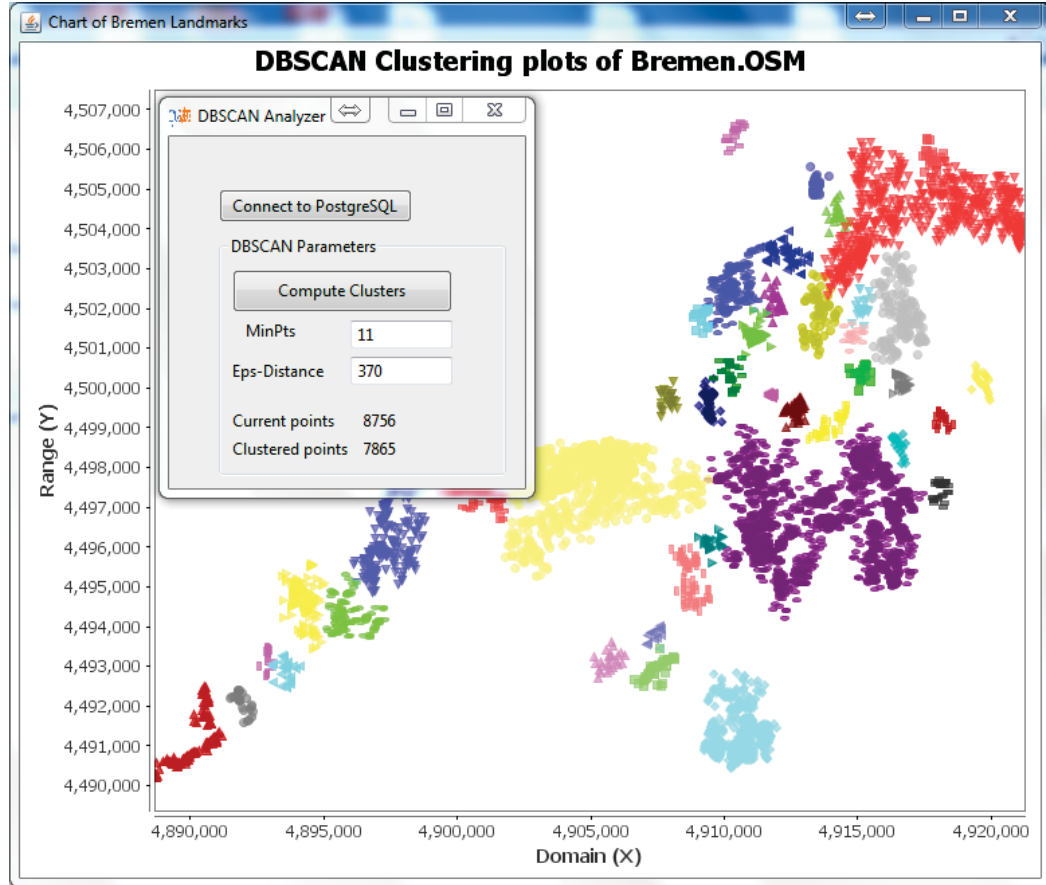
### 6.2 A Qualitative Spatial Layer

We abstract a qualitative spatial layer that covers three aspects: topology, direction, and distance using the `Qualitative_Rels_Abstracter()` function. The function applies Algorithm 1 presented in Section 4.3.

We have implemented the `Qualitative_Rels_Abstracter()` in PL/pgSQL. The function employs the topological predicates of PostGIS and the `Dir_Cones_Abstracter()`, `CDM_Abstracter()`, and `Abs_Dist_Abstracter()` to abstract the relations. In addition, `Qualitative_Rels_Abstracter()` inserts the abstracted spatial relations as well as their object pairs into `Abstracted_Rels` database table that is illustrated in Figure 6.2.

### 6.3 DBSCAN Clustering Implementation

In this section we report about the implementation of Density-Based Spatial Clustering of Applications with Noise (DBSCAN). DBSCAN has been described in Section 3.2 and 4.4. The implementation of DBSCAN consists of two parts: (1) The DBSCAN analyzer and (2) The DBSCAN spatial relations abstracter. We have implemented



**Figure 6.3:** A snapshot of DBSCAN analyzer; the user needs to specify two parameters: (1) the minimum number of points (*MinPts*) within a cluster and (2) the radius of a cluster (*Eps*).

the DBSCAN analyzer by using Java that we have connected with PostgreSQL. The DBSCAN analyzer allows for directly update the database table in PostgreSQL by assigning the generated cluster *id*'s to the corresponding objects in a database. The DBSCAN analyzer aims at clustering the database objects, visualizing them<sup>1</sup>, and analyzing them. A snapshot of the DBSCAN analyzer is depicted in Figure 6.3, in which the user can fill out the two parameters of DBSCAN: (1) the minimum number of points (*MinPts*) within a cluster and (2) the radius of a cluster (*Eps*). Next the user can press on “Compute Clusters” button and see the output of clustering. We have implemented the DBSCAN spatial relations abstracter in PL/pgSQL.

<sup>1</sup>Visualization is done via JFreeChart library <http://www.jfree.org/jfreechart/>

## 6. IMPLEMENTATION AND APPLICATIONS

---

The DBSCAN spatial relations abstracter consists of three functions:

1. The `Clusters_Enclosurer()`,
2. The `Clusters_Decisive_Rels_Abstracter()`,
3. The `Clusters_Rels_Abstracter()`.

`Clusters_Enclosurer()` generates the Minimum Bounding Rectangle (MBR), the Convex Hull (CH), and the ConCave Hull (CCH) for a set of points of a cluster. In particular, the MBR is generated by the `ST_Envelope`, the CH by `ST_ConvexHull`, and the CCH by `ST_ConcaveHull` (see Table 6.1). Moreover, we store the MBR, the CH, and the CCH of clusters in the `Clusterer_Clusters` database table that we depict in Figure 6.4. In turn, the `Clusters_Decisive_Rels_Abstracter()` uses the output of the `Clusters_Enclosurer()` as an input and computes the *decisive* and the *indecisive* relations between the generated shapes (e.g., the relations between the MBRs and the CCHs.) of clusters. We insert the spatial relations among clusters into the `Abstracted_QSRE_Among_Clusters` database table that we show in Figure 6.4.

Lastly, the `Clusters_Rels_Abstracter()` abstracts and stores the spatial relations within the clusters and between the clusters that are in *indecisive* relations. We record these spatial relations in `Abstracted_Rels_IN_Cluster` table that we depict in Figure 6.4.

### 6.4 Indexing Approaches Implementation

We have used Java and PL/pgSQL to implement five indexing approaches that are described in Chapter 5.

#### 6.4.1 A Hybrid Interpretation Tree and B<sup>+</sup>-Tree

We have implemented a Hybrid Interpretation Tree and B<sup>+</sup>-Tree (HITBT) indexing as the `HITBT_indexer()` function in PL/pgSQL. The `HITBT_indexer()` iterates over each attribute of the database tables (`Abstracted_Rels` and `Spatial_Objects` database tables in Figure 6.2) and constructs the B<sup>+</sup>-tree index on it. We create the B<sup>+</sup>-tree index in PostgreSQL by calling the following Data Description Language (DDL) command: `CREATE INDEX name ON table USING btree (column).`

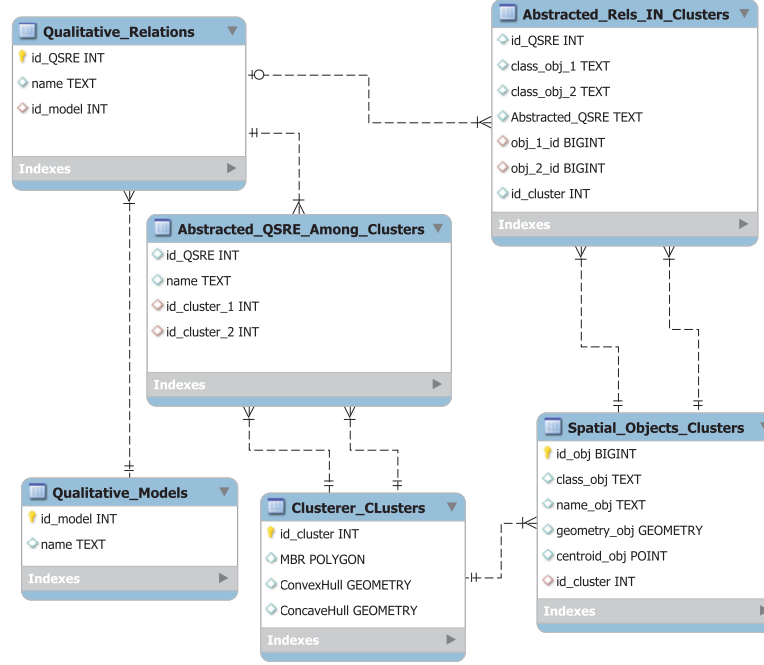


Figure 6.4: DBSCAN database schema design.

### 6.4.2 A Qualitative Hash Table Indexing

We have implemented a Qualitative Hash Table Indexing (QHTI) as the `QHTI_indexer()` function in PL/pgSQL. In particular, the `QHTI_indexer()` goes through each tuple of graph database ( $\mathcal{G}_D$ ), concatenates its attributes, and applies a linear hashing.

We construct the linear hash index in PostgreSQL by calling the following DDL command: `CREATE INDEX name ON table USING hash (column)`. The database schema of QHTI is depicted in Figure 6.5. We deploy the linear hash index on the `Merged_tuples` attribute of QHTI database table that is depicted in Figure 6.5.

It is worth mentioning that, the linear hash index is not well (or efficiently) implemented<sup>1</sup> in PostgreSQL v8.4 as well as the previous versions, which reduces the performance of the index. Nevertheless, we use PostgreSQL v9.0<sup>2</sup> as it offers a stable and well implemented linear hash index.

<sup>1</sup>According to <http://www.postgresql.org/docs/8.4/static/release-8-4.html>

<sup>2</sup>PostgreSQL: <http://www.postgresql.org/ftp/source/v9.0.0/>

## 6. IMPLEMENTATION AND APPLICATIONS

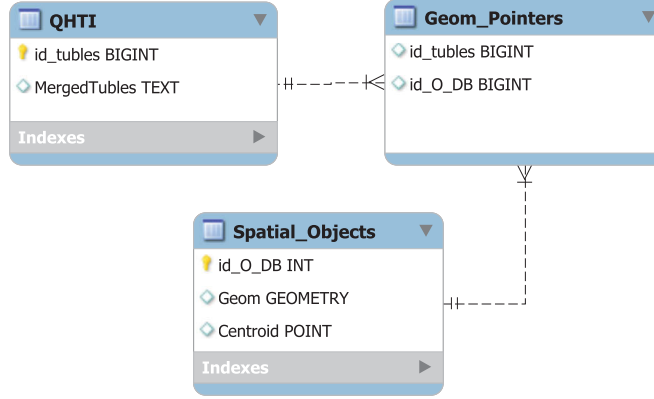


Figure 6.5: QHTI database schema design.

### 6.4.3 Qualitative Hash Table Compression

We have implemented a Qualitative Hash Table Compression (QHTC) as the `QHTC_indexer()` function in Java. Java is used due to the fact that QHTC requires complex data-structures to track the number of recurrences of QHTI database table tuples and their MBRs. The `QHTC_indexer()` first uses `HashMap`<sup>1</sup> data-structure offered by Java to track and represent recurring tuples uniquely and then to compute the MBRs of these recurrences. Next, the `QHTC_indexer()` stores the entities of `HashMap` in the QHTC database table (see Figure 6.6). In addition, the `QHTC_indexer()` stores the pointers of `HashMap` entries that point to geometric objects in the corresponding database tables (see the `QHTC_Pointers` and `Geom_Pointers` database tables in Figure 6.6). At this point, we apply linear hashing on the tuples of the QHTC database table using the following DDL command: `CREATE INDEX name ON table USING hash (column)`.

Lastly, the `QHTC_indexer()` constructs the  $B^+$ -tree index on geometric pointers stored in the `Geom_Pointers` database table to accelerate the retrieval of geometric objects.

### 6.4.4 The QHTC of Qualitative Models

Similar to QHTC, we have implemented the QHTC of Qualitative Models ( $QHTC^M$ ) as the `QHTCM_indexer()` function in Java. The `QHTCM_indexer()` iterates over each tuple of QHTI database table and aggregates the repeated spatial relations and stores them

<sup>1</sup>HashMap: <http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html>.

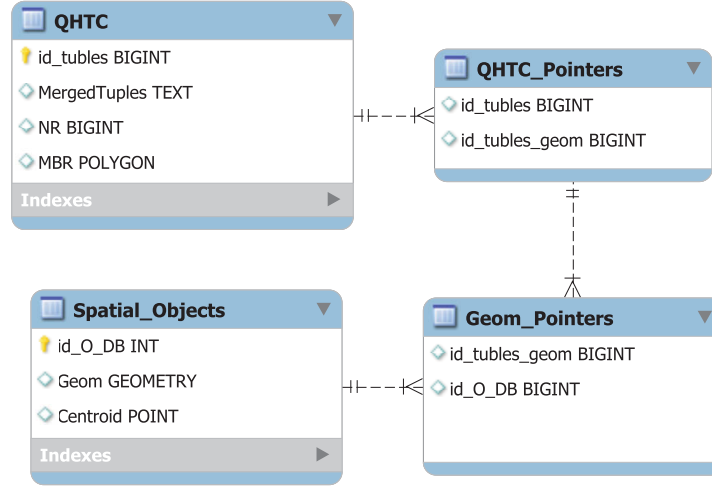


Figure 6.6: QHTC database schema design.

in the corresponding database table (see  $\text{QHTC}^M$ .SpRel database table in Figure 6.7). In addition, it applies a linear hashing on the spatial relations (SR) attributes by the calling the DDL command mentioned in Section 6.4.3. Afterwards it uses them to index object pairs which are stored in a separate database table (see  $\text{QHTC}^M$ .OPairs database table in Figure 6.7). Similar to  $\text{QHTC}$ , the  $\text{QHTC}^M$ .`indexer()` constructs the  $B^+$ -tree index on geometric pointers (in the `Geom_Pointers`) to speed-up the retrieval of geometric objects.

#### 6.4.5 The QHTC of Object Pairs

We have implemented the QHTC of Object Pairs ( $\text{QHTC}^P$ ) as  $\text{QHTC}^P$ .`indexer()` function in Java. The implementation of  $\text{QHTC}^P$ .`indexer()` is very similar to the one of  $\text{QHTC}^M$ , in the sense that the labels of object pairs that are used to index the spatial relations are now aggregated. Figure 6.8 shows that the database schema design which in turn is very similar to the one of  $\text{QHTC}^M$ .

## 6.5 Client-Side Interfaces

In this section we present interfaces that we call Client-Side Interfaces (CSIs). CSIs aim at enabling users to query the geo-spatial databases easily, intuitively, and qualitatively.

## 6. IMPLEMENTATION AND APPLICATIONS

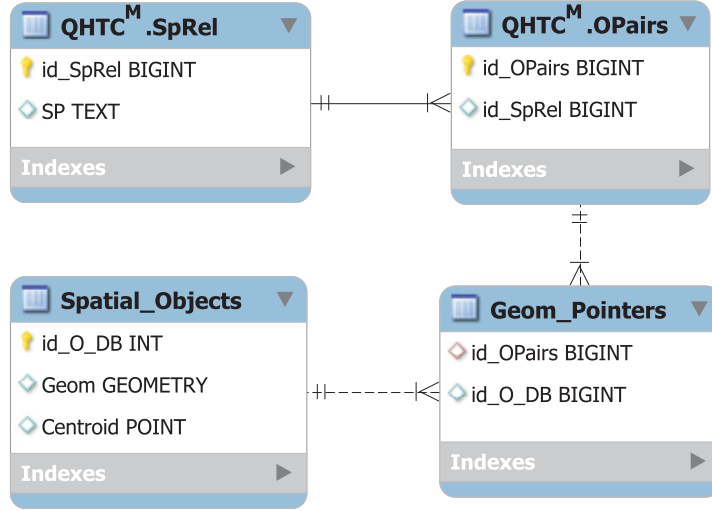


Figure 6.7: QHTC<sup>M</sup> database schema design.

As a testbed we have used a geo-referenced dataset of Bremen inner city that we have extracted from OpenStreetMap (OSM)<sup>1</sup> and contains 8756 objects (polygons).

The CSIs fall into two classes: Web-Based Interfaces (see Section 6.5.1) and Android-Based Interfaces (see Section 6.5.2).

### 6.5.1 Web-Based Interfaces

We have implemented the modules of Web-Based Interfaces (WBI) in HTML5, JavaScript, Asynchronous JavaScript and XML (AJAX), and PHP5. The extracted dataset has been stored in a PostgreSQL/PostGIS database, rendered by Osmarender, and viewed by OpenLayers.

In (Al-Salman et al., 2013a), we have developed a web application that called Qualitative Emergency Management System (QEMS). Similarly, here we instantiate a web application the-so-called Qualitative Spatial Management System (QSMS) from the WBI. Additionally, we supplement QSMS by the HITBT approach to speed-up the retrieval of geo-spatial data.

Figure 6.9 shows an overview of the QSMS architecture. In the first step Bremen dataset is rendered and projected by the Osmarender renderer. In the second step, the data is stored in the PostgreSQL/PostGIS database. In steps 3 and 4, the three types

<sup>1</sup>OpenStreetMap (OSM): <http://www.openstreetmap.org/>, is an open source and a collaborative project to create a free editable map of the world.



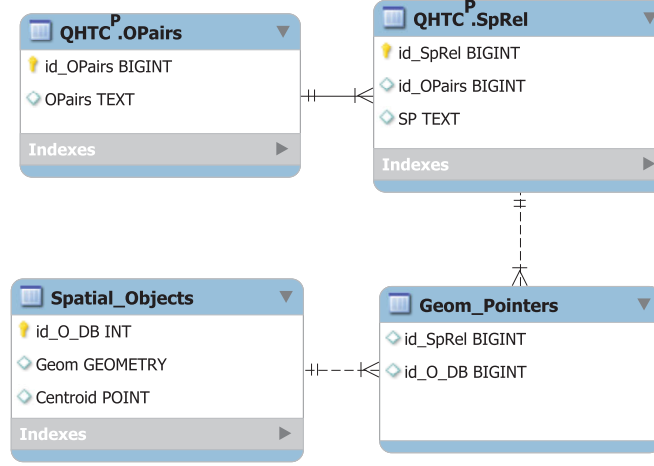
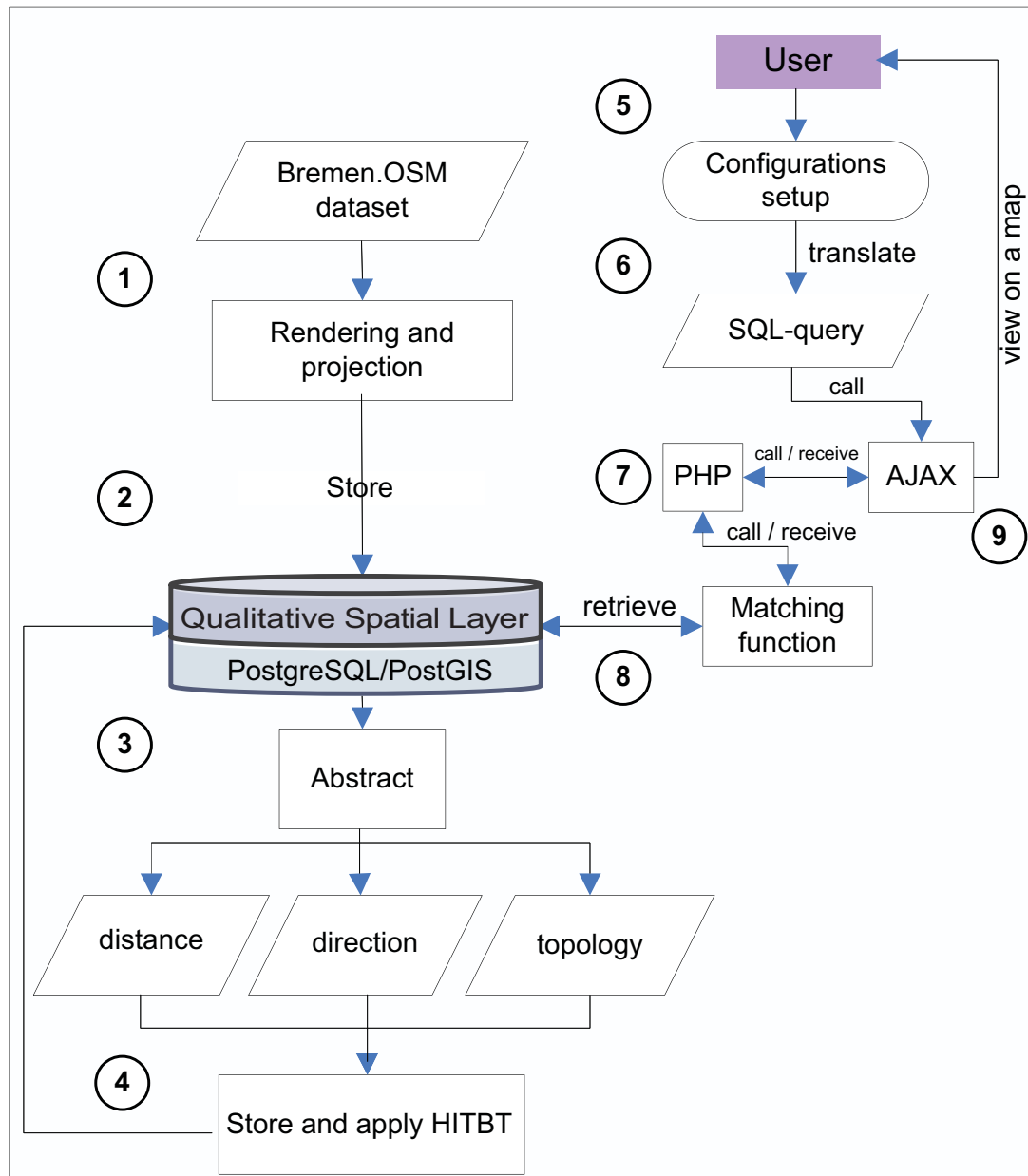


Figure 6.8: QHTC<sup>P</sup> database schema design.

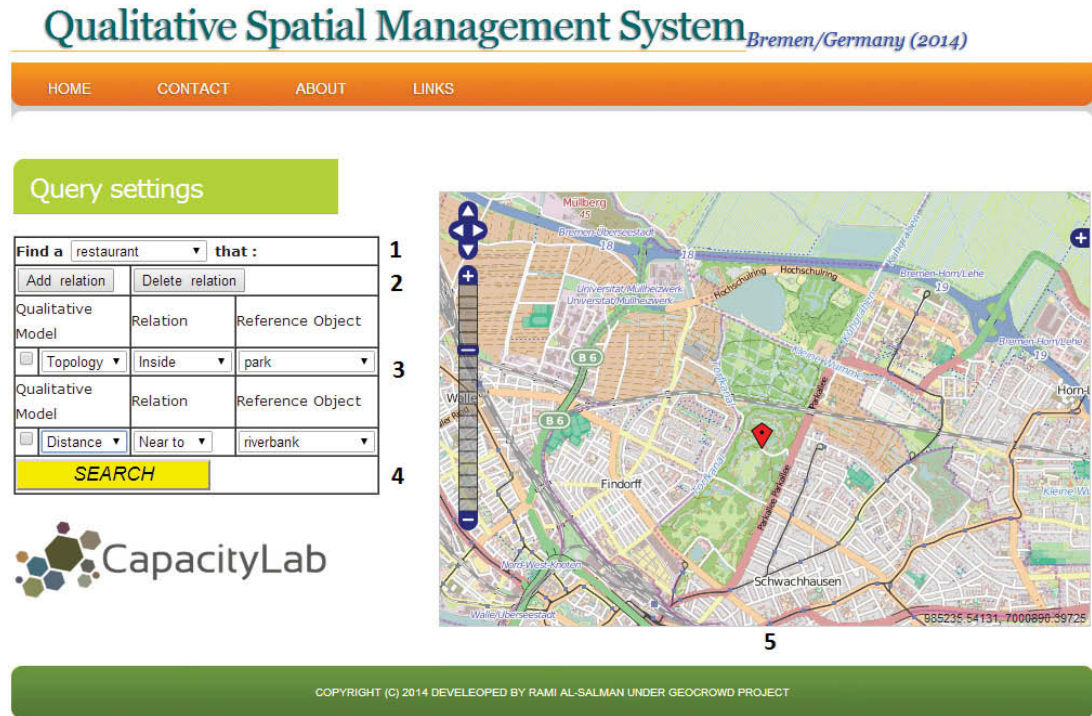
of qualitative spatial relations are abstracted and stored in the qualitative spatial layer upon the PostgreSQL/PostGIS database. In step 5, the user sets the spatial query configurations. These configurations are translated into Structured Query Language (SQL) in step 6. Accordingly, the SQL is sent via AJAX to PHP at the server side in step 7. In step 8, the matching function of HITBT (cf. Section 5.1.2) is applied. The PostGIS repository functions are called by PHP to retrieve a set of possible matches. In the last step, the results retrieved by the matching function are sent back via PHP to AJAX and presented on a map by OpenLayers<sup>1</sup>. We show a snapshot of the graphical user interface of QSMS in Figure 6.10. It illustrates that the output of such queries could be helpful for tourists who want to visit Bremen. As an example, it shows the answer to a query for a restaurant that is *inside* a park and *near* a riverbank. We marked different spots in the figure from 1 to 5. In the input field denoted by 1, a primary object (e.g., a restaurant) can be selected from a drop-down list. Directly below are objects which are supposed to be the reference object(s). These objects as well as their spatial relations can be added or deleted to/from the system (field 2). In connection to this object a qualitative spatial model (e.g., distance) and a qualitative relation (e.g., “*near*”) need to be selected (field 3). The user of the system does not need to specify any quantitative value for any qualitative spatial model. Next, an according query is generated (field 4). Based on this query, QSMS retrieves a set of matches which are displayed (as red markers) on a map (field 5). Red markers denote the result: a set of

<sup>1</sup>OpenLayers: <http://openlayers.org/>

## 6. IMPLEMENTATION AND APPLICATIONS



**Figure 6.9:** The system architecture of the Qualitative Spatial Management System.



**Figure 6.10:** A snapshot of the graphical user interface of QSMS.

matches that satisfy the spatial constraints that hold among pairs of objects in the user query.

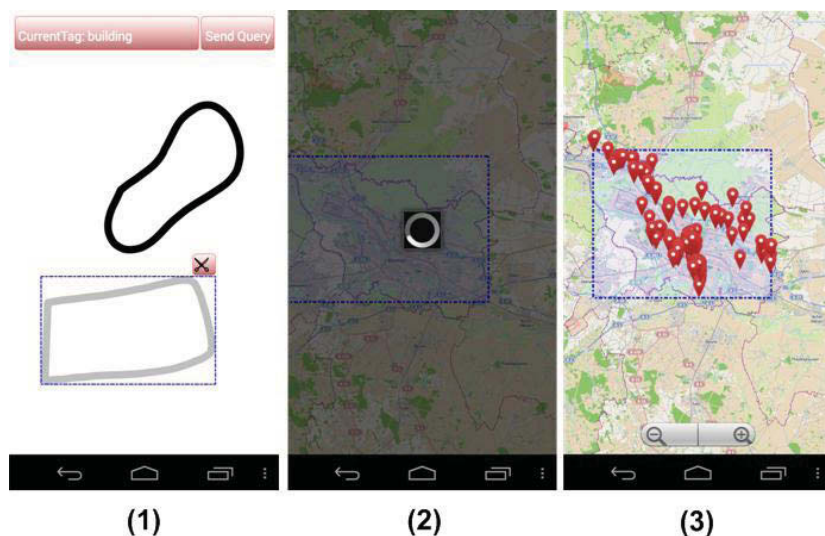
### 6.5.2 Android-Based Interfaces

We have implemented the modules of Android-Based Interfaces (ABIs) in Java/Android. ABIs aim to allow users to query the geo-spatial databases through Android smart mobile devices qualitatively and intuitively.

As an instantiation of ABIs, we have developed Android Sketching and Querying Tool (ASQT) that allows users to formulate their queries as sketch objects (e.g., lakes and streets) in an intuitive manner by means of gesture recognition libraries for Android smart-phones and tablets. Additionally, we apply Scalable Vector Graphic (SVG) and OpenGL libraries to allow for simple and smooth drawing. Using Java, we have implemented ASQT that runs on Android-based smart-phones and tablets. ASQT supports three kinds of gestures; (1) Selection, (2) Multi-touch zooming, and (3) Multi-touch rotation. It is worth mentioning that ASQT is based on the sketching and editing

## 6. IMPLEMENTATION AND APPLICATIONS

---



**Figure 6.11:** A snapshot of the Android Sketching and Querying Tool.

tool that we have proposed in (Al-Salman et al., 2013b).

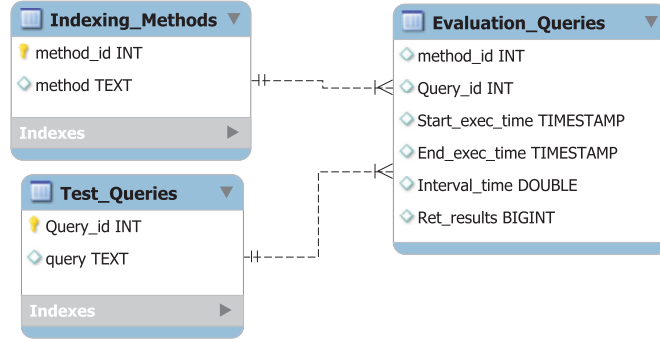
The system architecture of ASQT is very similar to QSMS one, except that we now use Java instead of PHP to call PostGIS functions. Additionally, we use the QHTC matching approach presented in Section 5.3.2 in order to accelerate the query processing.

A snapshot of the graphical user interface of ASQT is depicted in Figure 6.11. A user can select any drawn object by pressing it. With two or three finger gestures on the Minimum Bounding Rectangle (MBR) (Figure 6.11(1)) objects can be rotated and zoomed. Moreover, the user can delete the selected object easily by pressing on the **x** button at the right corner of the MBR. In order to submit the queries, the user needs to annotate the drawn objects and press “Send Query”. Next, the geometries of the query are sent to a server that computes the spatial relations among them and matches them against the spatial relations of the qualitative spatial layer.

Figure 6.11(2) shows the benefit of using QHTC, where the MBR approximation of the matchings is retrieved in an early stage of the matching process. Lastly, the actual centroids of matchings are retrieved and depicted on a map (Figure 6.11(3)).

### 6.6 System Evaluation

System evaluation is done by recording the execution time and the number of retrieved results of queries. Two kinds of queries can be processed: (1) user-defined queries and



**Figure 6.12:** System evaluation database schema design.

(2) system-defined queries. The former is naturally provided by users of a system. The latter is essential for evaluation purposes and operated by `Auto_Queries_Generator()`. We have implemented the `Auto_Queries_Generator()` in PL/pgSQL and it is able to randomly generate spatial queries. Finally, we record the queries, their execution time, and their matching approaches in the corresponding database tables that we show in Figure 6.12. Algorithm 21 shows the steps for generating queries by the `Auto_Queries_Generator()`. The algorithm first iterates over each tuple of  $\mathcal{G}_D$  and uses it to fetch the corresponding entry from the  $T^C$  using the `Search_QHTC()` described in Section 5.3.2 (lines 2 and 3). The main goal of specifically fetching the entry of  $T^C$  of QHTC is to detect the number of recurrences ( $NR$ ), i.e., the number of matches to the generated query. This demonstrates another benefit of using the QHTC approach. When  $NR$  is between  $Min$  and  $Max$  values, then the  $Unq$  value is checked (lines 4 and 5). Now, if the generated queries are specified to be unique (line 5), then the repository of queries (Test\_Queries database table in Figure 6.12) is checked to make sure that the query does not exist in the repository (line 6). When a query is unique, then the corresponding queries (e.g., queries of HITBT and QHTI) are generated based on this query and stored in Test\_Queries (line 7). On the other side, the queries are directly generated and stored without checking the Test\_Queries repository when they are allowed to be non-unique or repeated (lines 10 to 13). Finally, the procedure stops constructing queries when the number of the generated queries exceeds  $nq$  (lines 15 to 17). We note that the `Auto_Queries_Generator()` only generates queries without executing them. We will execute the queries and record their execution time in the next chapter for evaluation purposes.

## 6. IMPLEMENTATION AND APPLICATIONS

---



---

**Algorithm 21:** `Auto_Queries_Generator`( $\mathcal{G}_{\mathcal{D}}$ ,  $T^C$ ,  $Min$ ,  $Max$ ,  $Unq$ ,  $nq$ )

---

**output:** `Test_Queries`: a number of generated queries stored in the database table

---

```

1 initialization:  $Match \leftarrow \text{NULL}$ ;  $c \leftarrow 0$ ;  $\ell \leftarrow |\mathcal{G}_{\mathcal{D}}|$ ;
2 for  $k \leftarrow 1$  to  $\ell$  do
3    $Match \leftarrow \text{Search\_QHTC}(T^C, \mathcal{G}_{\mathcal{D}}[k])$ ;
4   if  $Match.NR > Min \wedge Match.NR < Max$  then
5     if  $Unq == \text{TRUE}$  then
6       if Test_Queries does not contain  $\mathcal{G}_{\mathcal{D}}[k]$  then
7         Test_Queries.Insert( $c$ , GenerateQueries( $\mathcal{G}_{\mathcal{D}}[k]$ ));
8          $c \leftarrow c++$ ;
9       end
10    else
11      Test_Queries.Insert( $c$ , GenerateQueries( $\mathcal{G}_{\mathcal{D}}[k]$ ));
12       $c \leftarrow c++$ ;
13    end
14  end
15  if  $c \geq nq$  then
16    EXIT ;
17  end
18 end

```

---

### 6.7 Summary

In this chapter we have described the components of our system that we have called **QualEnabler**. In each component, we have explained the corresponding database design schema.

In Section 6.1, we have explained the functions that have been implemented and used to integrate the directional and distance models into PostGIS.

Next, in Section 6.2 we have described the functions that have been used to abstract the qualitative spatial layer into PostgreSQL.

Section 6.3 has elaborated on the requirements and functions that have been needed to implement the DBSCAN approach. We have integrated the DBSCAN analyzer and abstracter into PostgreSQL. Thus, it is possible directly to visualize the point clusters of databases and abstract the spatial relations within and among the clusters of databases.

Afterwards, we have described the data-structures, functions, and commands that were essential to implement our indexing approaches (Section 6.4). Accordingly, we have integrated the indexing approaches into PostgreSQL.

In turn, Section 6.5.1 has demonstrated the client-side applications that have been implemented based on the aforementioned components.

Lastly, Section 6.6 has explained the evaluation and testing unit of `QualEnabler`. It has presented an important function that has been called the `Auto_Queries_Generator()` and will be used in the next chapter to evaluate our approaches.

## 6. IMPLEMENTATION AND APPLICATIONS

---



## Empirical Evaluation

In this chapter, we report about experiments on real-world and synthetic datasets to evaluate our work. We have carefully selected the datasets to cover as many cases as possible. We carried out experiments on real-world dataset to evaluate two major aspects: (1) the space reduction of the qualitative data and (2) the processing time of the proposed matching approaches.

In the first aspect, we evaluate the ability of the density-based spatial clustering of applications with noise approach introduced in Chapter 4 to reduce the qualitative data (see Section 7.1). Next, we examine the possibility of reducing qualitative data by the three approaches proposed in Chapter 5: (1) Qualitative Hash Table Compression (QHTC), (2) QHTC of Qualitative Models (QHTC<sup>M</sup>), and (3) QHTC of Object Pairs (QHTC<sup>P</sup>) (see Section 7.2.2). In the second aspect, we evaluate the performance and scalability of the seven matching approaches presented in Chapter 4 and 5 (see Section 7.2).

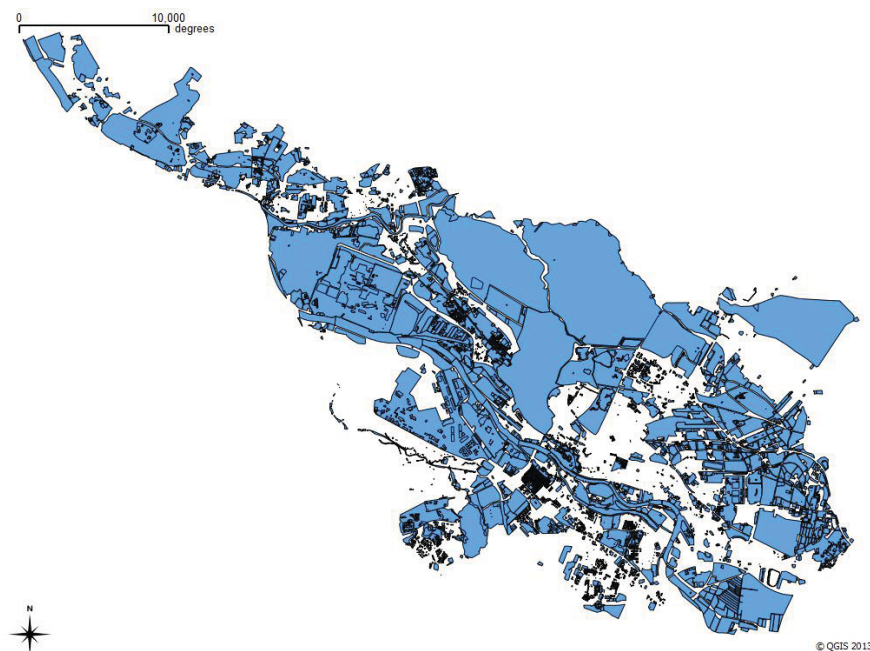
For the same reasons and motivation, we perform the aforementioned evaluations on a synthetic dataset (see Section 7.3).

### 7.1 Clustering Experiments

In this section, we present experiments and evaluate the ability of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to reduce the spatial relations of the graph database ( $\mathcal{G}_D$ ). The reduction process occurs in two stages: Filtering Clustering Candidates (see Section 7.1.2) and Selecting Clustering Candidate (see Section 7.1.3).

## 7. EMPIRICAL EVALUATION

---



**Figure 7.1:** A snapshot of the real dataset of Bremen inner city.

We start by giving the experimental settings and present the experimental results and findings afterwards.

### 7.1.1 The Experimental Settings of Clustering

In our experiments, we have used the OpenStreetMap (OSM)<sup>1</sup> geo-referenced real dataset of Bremen inner city that has been stored in the PostgreSQL/PostGIS database and contained 8756 objects. Originally, we have extracted 9000 objects from OSM. However, not all the extracted objects are annotated by volunteers. Hence we apply a preprocessing step to eliminate the non-annotated objects. Consequently, 244 objects are eliminated and 8756 objects are left. Figure 7.1 exhibits a snapshot of the inner city of Bremen constructed based on the objects of our OSM dataset and viewed by Quantum-GIS<sup>2</sup>.

We selected Bremen dataset for three reasons: (1) most of its objects are annotated by volunteers, (2) it has a high quality in terms of annotated objects (as most urbanized and populated cities in Germany) (Ludwig et al., 2011; Zielstra and Zipf, 2010), and

---

<sup>1</sup>OpenStreetMap (OSM): <http://www.openstreetmap.org>.

<sup>2</sup>Quantum-GIS (QGIS): <http://www.qgis.org/en/site/>.

**Table 7.1:** DBSCAN Parameter settings.

parameter	setting
The minimum points within a cluster ( <i>MinPts</i> )	2, 3, 4, ..., 11
The radius of a cluster ( <i>Eps</i> ) measured by meters	50, 60, 70, ..., 490

(3) it covers various densities (sparse/dense) and distributions (normal/non-normal) of areas and object types (Zielstra and Zipf, 2010).

DBSCAN uses two parameters: (1) the minimum number of points (*MinPts*) within a cluster and (2) the radius of a cluster (*Eps*). We accordingly use them to examine the performance of DBSCAN, as Table 7.1 shows the parameters with their possible values. In each experiment, we vary the *MinPts* parameter, then for each *MinPts* value, we iterate over the *Eps* values.

### 7.1.2 Filtering Clustering Candidates

As with any data mining process, selecting the values for the *MinPts* and the *Eps* is a critical and non-trivial process. Our goal of clustering is to save the spatial relations of  $\mathcal{G}_{\mathcal{D}}$  as much as possible, which can be arguably achieved by the following two procedures:

1. minimizing the number of outliers (by maximizing the number of clustered objects),
2. generating equal-sized clusters so that each cluster contains the same (or nearly the same) number of objects as the other clusters.

Therefore, we define two criteria to select the cluster candidates: (1) the number of outliers and (2) the maximum cluster size that gives the good indications about the balancing of the size of clusters.

We start by setting the *MinPts*=2 and then we iterate over all the *Eps* values from 50 to 490 incremented by 10.

DBSCAN terminates the clustering process when two conditions are satisfied: (1)  $\geq 50\%$  of the database objects are clustered and (2) a large merge occurs between two (or possibly more) clusters. A large merge is indicated when we target a merge among clusters that lead to (more than) double the maximum size cluster and an imbalance in the number of objects in the clusters compared to others. For simplicity, we may denote the clustering by DBSCAN using its parameters as  $\text{DBSCAN}(\text{MinPts}, \text{Eps})$ .

## 7. EMPIRICAL EVALUATION

---

Testing all the possible values for the *MinPts* and the *Eps* is an expensive process. Other possibilities include hierarchical approach for selecting the *Eps* values for each *MinPts*. For example, if we know that the maximum size cluster is (more than) doubled for *Eps* values between 250 and 350, then we only need to search *Eps* values  $< 350$ , whereas the *Eps* values  $> 350$  can be safely neglected. Nevertheless, developing optimization approaches to detect the peak situations is beyond the scope of this dissertation.

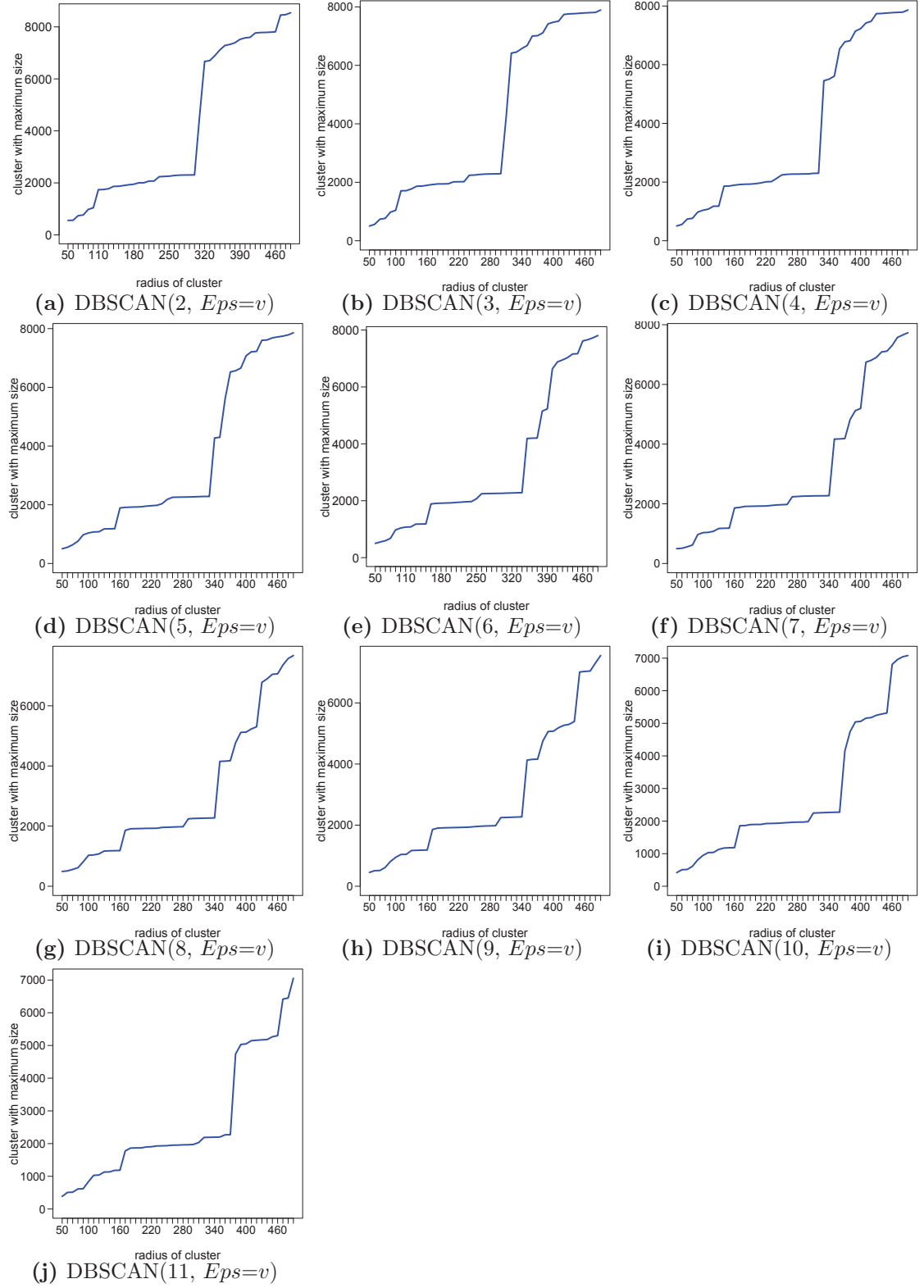
Figure 7.2 depicts the peaking situations for the maximum cluster size for the *MinPts*=2 to 11. For instance, Figure 7.2(a) shows that the maximum size of clusters peaks when  $Eps > 300$ . In more detail and clarifications, Figure 7.3 illustrates several snapshots of the DBSCAN clustering, in which the *MinPts* is fixed at 2 and several *Eps* values are tested. Figure 7.3(a) particularly exhibits that only few points are clustered when the  $Eps=50$ . The number of clustered objects is increased with the *Eps* (Figure 7.3(b), 7.3(c), and 7.3(d)). A large merge happens when the *Eps* becomes more than 300 (directly at 310). This indicates that the clustering should be terminated at this point (Figure 7.3(e)). In fact, the maximum cluster size jumps from 2307 to 4565 (its size is almost doubled compared to its predecessor) when the  $Eps = 310$ . Eventually, most of the objects are included in a single cluster when the *Eps* value reaches 490 (Figure 7.3(f)). We perform the same process by using the *MinPts* values from 3 to 11 (Figure 7.2(b) to 7.2(j)) in order to select the remaining clustering candidates. As exhibited in Figure 7.2(b) to 7.2(j), increasing the *MinPts* requires either increasing or keeping the same *Eps* of its predecessor in order to peak the maximum size of clusters.

In Table 7.3, we select and list the values for the pairs (*MinPts*, *Eps*) that act as the 10 clustering candidates. Moreover, we calculate the clustering execution time, the number of clusters, the maximum cluster size, and the number of outliers. In all the experiments, DBSCAN took between 12 and 15 seconds to cluster the objects.

In order to measure the strength and significance of the relationship between pairs of variables (e.g., (*MinPts*, the number of clusters)), we use Pearson correlation coefficient ( $r$ ) and  $p$ -value respectively (Stigler, 1989). The formula of  $r$  is defined as follows:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (7.1)$$

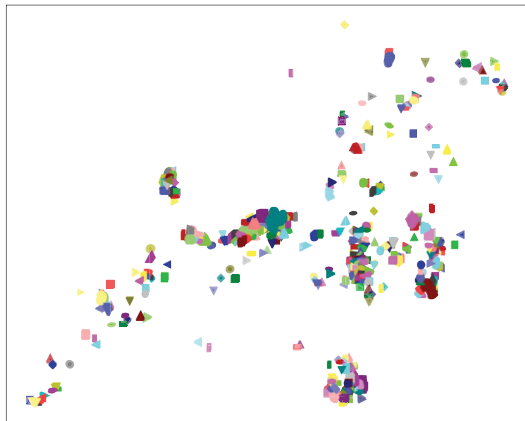
## 7.1 Clustering Experiments



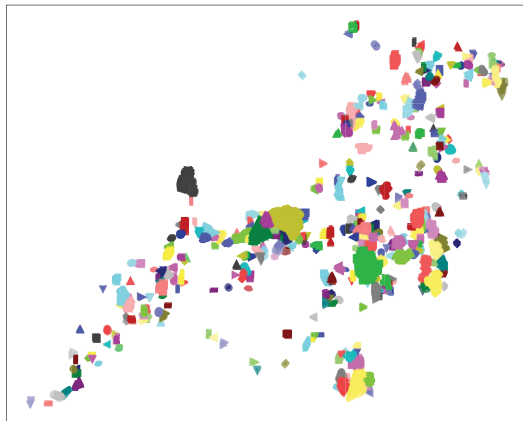
**Figure 7.2:** Snapshots of DBSCAN( $MinPts$ ,  $Eps=v$ ),  $v$ : the radius of clusters varied from 50 to 490 meters incremented by 10 meters.

## 7. EMPIRICAL EVALUATION

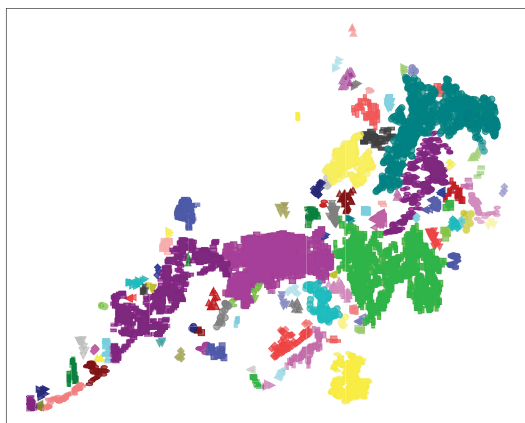
---



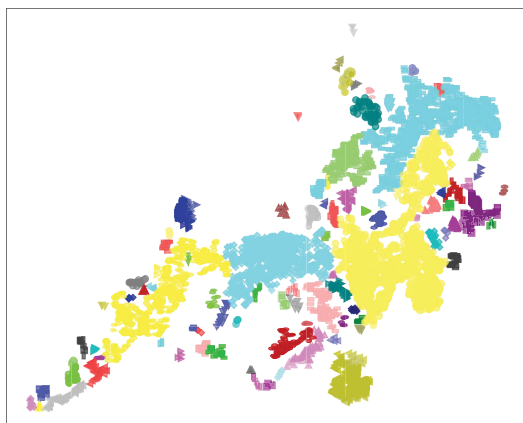
(a) DBSCAN( $MinPts=2$ ,  $Eps=50$ )



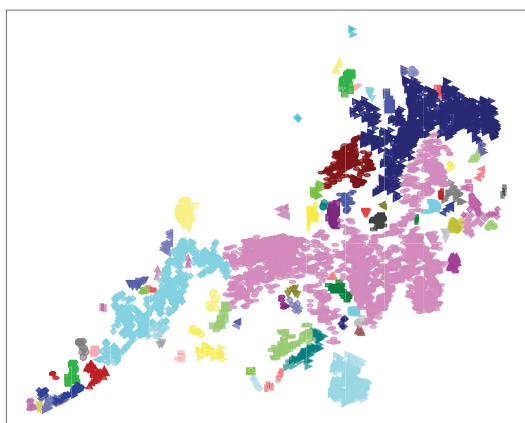
(b) DBSCAN( $MinPts=2$ ,  $Eps=100$ )



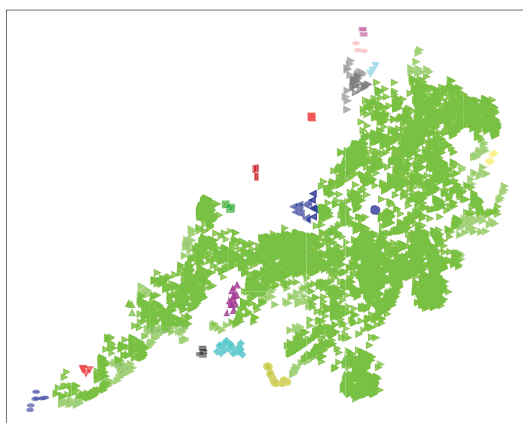
(c) DBSCAN( $MinPts=2$ ,  $Eps=290$ )



(d) DBSCAN( $MinPts=2$ ,  $Eps=300$ )



(e) DBSCAN( $MinPts=2$ ,  $Eps=310$ )



(f) DBSCAN( $MinPts=2$ ,  $Eps=490$ )

**Figure 7.3:** Snapshots of DBSCAN( $2$ ,  $Eps$ ):  $Eps \in \{50, 100, 290, 300, 310, 490\}$ .

where  $n$  is the number of observations,  $x_i$  and  $y_i$  are individual observations and  $\bar{x}$  and  $\bar{y}$  are the means for variables  $x$  and  $y$  respectively. Given  $-1 < r < +1$ ,  $r$  conveys the correlation of pairs of variables, so the larger the correlation (the closer to 1), the stronger the relationship.

In turn, we obtain  $p$ -value based on the  $t$ -distribution of  $r$  (see Equation 7.2) and based on Fisher's  $r$ -to- $z$  transformation (Stigler, 1989).

$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}} \quad (7.2)$$

Given a  $t$ -distribution with  $k$  degrees of freedom for a test statistic. We compute the  $p$ -value for a two-tailed test as:

$$p - value = 2 * P(t_k > |t_{stat}|) \quad (7.3)$$

where  $P$  is the probability of null hypothesis that the coefficient is 0 and  $|t_{stat}|$  is the absolute value of the calculated test statistic. In contrast to  $r$ , given  $0 < p < +1$ , the lower  $p$ -value indicates that the relationship of pairs of variables is statistically more significant than the higher one. The main goal of using  $p$ -value is to support the correlation that can be obtained from  $r$ . According to (Manktelow and Chung, 2004), the strength of the correlation between any two variables can be verbally described for a value of  $r$ <sup>1</sup> as follows:

- 0.00 “no correlation ”
- 0.00 to  $\pm 0.19$  “very weak”
- $\pm 0.20$  to  $\pm 0.39$  “weak”
- $\pm 0.40$  to  $\pm 0.59$  “moderate”
- $\pm 0.60$  to  $\pm 0.79$  “strong”
- $\pm 0.80$  to  $\pm 1.0$  “very strong”
- $\pm 1.0$  “perfect”

---

<sup>1</sup>A value of  $r$  is always ranged between two values that have the same sign as  $r$ .

## 7. EMPIRICAL EVALUATION

---

**Table 7.2:**  $r$  and  $p$ -value for  $MinPts$  with other variables.

Measure		$Eps$	# of clusters	# of outliers
$r$	$MinPts$	0.953	-0.852	0.984
$p$ -value	$MinPts$	0.00002	0.0018	0.0000004

**Table 7.3:** The clustering candidates of DBSCAN experiments.

$MinPts$	$Eps$	Exec. time[sec]	# of clusters	Maximum cluster size	# of outliers
2	300	12	90	2307	258
3	300	12	81	2293	301
4	320	12	61	2301	410
5	330	12	52	2284	489
6	340	14	52	2283	537
7	340	13	48	2273	689
8	340	15	47	2272	785
9	340	12	50	2271	849
10	360	12	47	2273	832
11	370	12	43	2273	890

Regarding  $p$ -value, the relationship between two variable is significant if  $p$ -value  $\leq 0.05$  (Manktelow and Chung, 2004).

As pairs of variables of interest, Table 7.2 shows the correlation coefficient  $r$  and  $p$ -value between the  $MinPts$  and other variables. It conveys that the correlation coefficient between the two pairs ( $MinPts$ ,  $Eps$ ) and ( $MinPts$ , the number of outliers) is significant, *positive*, “very strong”, and linear. The correlation coefficient between the ( $MinPts$ , the number of clusters) is significant, *negative*, “very strong”, and linear.

### Discussion:

In the filtering phase, we selected the clustering candidates that have the minimum number of outliers as well as the semi equal-sized clusters. In particular, reducing the number of outliers (see Figure 7.3) is achieved by increasing the  $Eps$  value as much as possible for a given  $MinPts$ . By increasing the  $Eps$  value, it allows the inclusion of more objects in clusters that results in decreasing the number of outliers. On the other side, generating the semi equal-sized clusters are done by preventing the possibility of larger clusters being merged. This is due to the fact that merging clusters leads to an imbalance in the number of objects in the clusters compared to other clusters.



The large merge is detected based on the maximum cluster size compared to its predecessor. For a given  $MinPts=2$  (see Figure 7.3(e)), at least a large merge occurs between two clusters when the  $Eps$  is varied to 310. This merge is detected since the maximum cluster size is almost doubled compared to its predecessor (see Figure 7.3(d)).

The merge situations for the  $MinPts = 3$  to 11 (Figure 7.2(b) to 7.2(j)) can be validated as well. Except the eighth and ninth clustering candidates (DBSCAN(9, 340) and DBSCAN(10, 360)), the number of outliers of the selected candidates (see Table 7.3) increased with the number of the  $MinPts$ . On the contrary, the number of clusters either decreased or stayed the same with a decrease in the number of the  $MinPts$ . By increasing the  $MinPts$  restricts including the objects in the clusters, that implies a smaller number of clusters and a higher number of outliers.

Regarding the clustering by DBSCAN(9, 340) that can be considered as an exception, since the number of clusters is increased to exceed its two predecessors. The  $Eps$  value is equal to its two successors. Additionally, the number of outliers has increased more than its successor ( $MinPts=10$ ).

We determined that when  $MinPts=9$  and  $Eps=340$  are used by DBSCAN, it may prevent some clusters from being created when their  $MinPts < 9$ . Consequently, their objects become available for other clusters that most likely need few objects to complete their number of objects ( $MinPts \geq 9$ ).

### 7.1.3 Selecting Clustering Candidate

In the previous section, we generated the clustering candidates based on their maximum cluster size and their numbers of outliers. Here, we take one step further by selecting the clustering candidate based on their ability to reduce the spatial relations.

In Chapter 4, we indicated that the Minimum Bounding Rectangles (MBRs) should be adequately used to compute the distance and directional relations, whereas the ConCave Hulls (CCHs) should be used to compute the topological relations. Therefore, we use the MBRs and the CCHs representations to compute the spatial relations within and among clusters. We also pointed out that the CCH takes a parameter ( $\alpha$ ) that allows a polygon (cluster) to shrink with a certain amount of its original size. We decrease and vary the  $\alpha$  value from 1 to 0.4 with 0.1 step for all the candidates, where 1 denotes the convex hull (0% i.e., shrinking is not allowed) and 0.4 denotes that 60 % percent of the original

## 7. EMPIRICAL EVALUATION

---

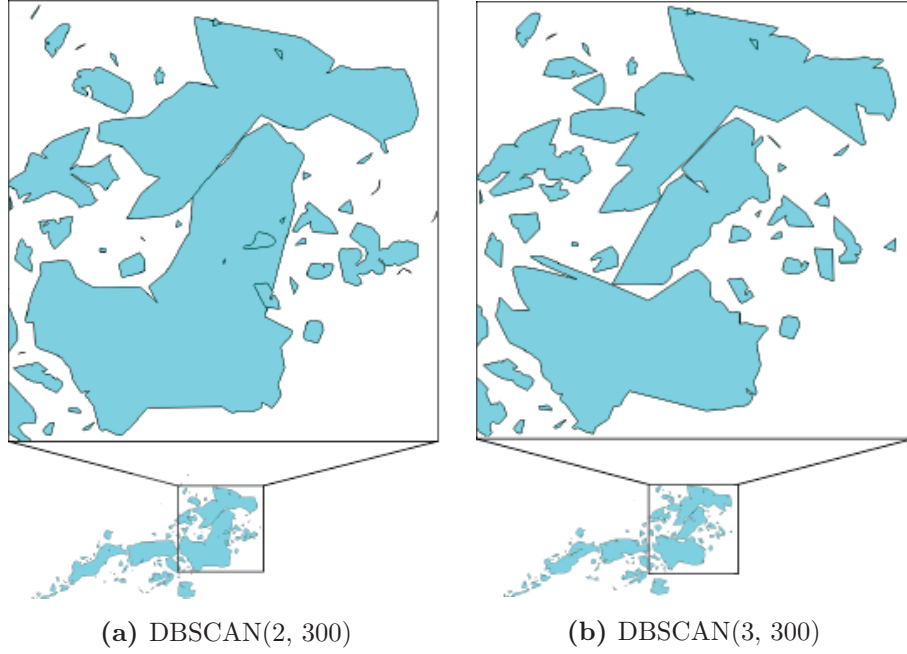
**Table 7.4:** Generating the CCHs of clusters using  $\alpha$ .

$MinPts$	$Eps$	Exec. Time[sec]	Non- <i>disjoint</i> CCH Rel.	Non- <i>disjoint</i> CH Rel.	Non- <i>disjoint</i> MBR Rel.	# of Relations	$\alpha$
2	300	9.9	18	60	130	8010	0.9
3	300	20.2	6	46	108	6162	0.8
4	320	8.7	6	30	104	3660	0.9
5	330	7.8	4	26	74	2652	0.9
6	340	10.1	10	24	60	2652	0.9
7	340	10.7	6	14	46	2256	0.9
8	340	9.8	6	18	46	2162	0.9
9	340	11.8	6	20	44	2450	0.9
10	360	12.4	4	14	44	2162	0.9
11	370	11.5	6	18	48	1806	0.9

size of the polygon can be shrunk. Afterwards, we select the  $\alpha$  value that leads to the least number of the non-*disjoint* topological relations (e.g., *inside* and *covers*).

By using  $CCH(\alpha)$ , we denote the process of enclosing a cluster using  $\alpha$  parameter. The selection of the CCHs over the MBRs and the CHs to compute the topological relations is validated by the results depicted in Table 7.4, where the CCHs produce less number of non-*disjoint* relations than others. It indicates that the most appropriate  $\alpha$  values of CCH are 0.8 for DBSCAN (3, 300) and 0.9 for the other candidates. It also shows that computing the  $CCH(0.8)$  requires almost double time as compared to  $CCH(0.9)$ .

Note that we still did not select the clustering candidate. In order to select the clustering candidate, we compare the candidates with respect to the space reduction rates of the qualitative spatial relations. Based on the definition of the *decisive* relation described in Section 4.4.1, we calculate the number of spatial relations per qualitative aspect that can be saved by each clustering candidate. In other words, we calculate the number of spatial relations that do not need to be stored in  $\mathcal{G}_D$  (or qualitative spatial layer). In particular, calculating the *decisive* relations involves both the outliers and clusters. At this point, the outliers are treated as clusters as well. We compute the number of *decisive* relations between the clusters themselves as well as between the outliers and the clusters. Table 7.5 exhibits the reduction rates of topological, directional, and distance relations as well as their average reduction. In particular, the first candidate DBSCAN(2, 300) saves up to 79.28 %, 28.60 %, and 7.76 % of



**Figure 7.4:** A snapshot of the CCHs: DBSCAN(2, 300) v.s. DBSCAN(3, 300).

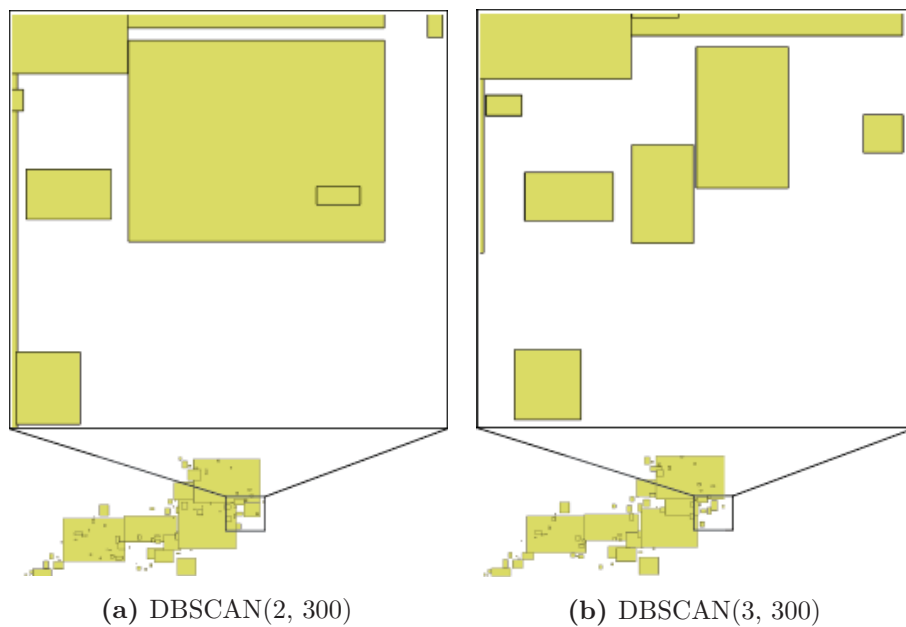
the topological, directional, and distance relations, respectively. The next candidate DBSCAN(3, 300) raises the reduction rates to be 85.66 %, 33.80 %, and 8.89 % of topological, directional, and distance relations, respectively. Nevertheless, there is no perfectly positive or negative relationship between the reduction rates and the *MinPts* or *Eps*. For example, increasing the *MinPts* does not necessarily lead to an increase in the reduction rates (e.g., the topological reduction rates of DBSCAN(4, 320), which are less than the ones of DBSCAN(3, 300)).

There appears to be a positive correlation between the *MinPts* and the average reduction rate. In particular, the correlation coefficient  $r$  and  $p$ -value between the *MinPts* and the average reduction rate are 0.889 and 0.0005, respectively. The  $r$  value indicates that they are “very strongly”, and linearly correlated, while  $p$ -value gives an evidence that the relationship is statistically significant.

Next, we select the clustering candidate that will give the highest reduction rates. Regarding the topological relations, DBSCAN(3, 300) can be considered as the candidate that offers the highest reduction. In turn, DBSCAN(11, 370) can be viewed as the candidate that provides the highest reduction rate for the distance and the directional relations. Finally, DBSCAN(11, 370) is the candidate that finds the highest

## 7. EMPIRICAL EVALUATION

---



**Figure 7.5:** A snapshot of the MBRs: DBSCAN(2, 300) v.s. DBSCAN(3, 300).

**Table 7.5:** The reduction rates for the topological, directional, and distance relations as well as their average reduction.

<i>MinPts</i>	<i>Eps</i>	Topology Red.	Direction Red.	Distance Red.	Avg Red.
<b>2</b>	300	79.28 %	28.60 %	7.76 %	38.55 %
<b>3</b>	300	85.66 %	33.80 %	8.89 %	42.78 %
<b>4</b>	320	84.24 %	27.09 %	9.45 %	40.26 %
<b>5</b>	330	84.07 %	27.58 %	7.94 %	39.86 %
<b>6</b>	340	85.01 %	37.38 %	12.82 %	45.07 %
<b>7</b>	340	85.41 %	38.44 %	12.98 %	45.61 %
<b>8</b>	340	85.54 %	39.09 %	13.96 %	46.20 %
<b>9</b>	340	85.60 %	40.01 %	15.06 %	46.89 %
<b>10</b>	360	85.62 %	39.93 %	14.58 %	46.71 %
<b>11</b>	370	85.47 %	40.60 %	16.23 %	47.43 %

average reduction rate. After gathering this information, we decide to select a clustering candidate that has the highest average reduction rate, since none of the candidates provided the highest reduction rate to all kinds of spatial relations. Accordingly, we select DBSCAN(11, 370) to abstract the *decisive* relations. Next, we abstract the spatial relations between the all the object pairs of all pairwise *disjoint* clusters that are in *indecisive* relations.

### Discussion:

In the selection phase, we evaluated the ability of all clustering candidates to reduce the three types of spatial relations and to select the clustering candidate. We used the MBRs to abstract the distance and directional relations, and the CCH to abstract the topological relations. As shown in Table 7.4, computing the topological relations using the CCH of clusters leads to a smaller number of the non-*disjoint* relations than the MBRs and CHs. This occurs because the enclosure of the points cluster using CCH is usually more tighter than the MBRs and CHs. It also shows that computing the CCH(0.8) requires almost double the time as compared to the CCH(0.9). This is due to the fact that the CCH(0.8) takes more time to shrink 20 % of the polygons (clusters) sizes, whereas the CCH(0.9) only needs to shrink 10 % of the polygons sizes.

The reduction rate of qualitative relations differs from one to another clustering candidate. Consider DBSCAN(2, 300) and DBSCAN(3, 300) (see Table 7.5) as examples. Regarding the topological relations, DBSCAN(3, 300) has a reduction rate greater than DBSCAN(2, 300) for two reasons.

First, DBSCAN(3, 300) generates smaller CCHs than those produced by DBSCAN(2, 300) that leads to increase the number of *disjoint* relations among the clusters. This can be more justified by the regions of interest of the clusters depicted in Figure 7.4 that shows the topological relations among the CCHs. In particular, it conveys that the CCHs generated by DBSCAN(2, 300) are large enough to include other CCHs, which implies less *decisive* relations and thus a smaller reduction rate. On the other hand, DBSCAN(3, 300) produces smaller CCHs than those produced by DBSCAN(2, 300). Thus, a smaller number of the CCHs are included (see Figure 7.4) in others which implies more *decisive* relations and thus a higher reduction rate.

Second, DBSCAN(3, 300) is able to enclose its clusters using the CCHs more tightly than DBSCAN(2, 300) as well as other candidates. This explains why DBSCAN(3,

## 7. EMPIRICAL EVALUATION

---

300) has the highest reduction rate for the topological relations in comparison to other candidates.

Similarly, the reduction rates of the directional and the distance relations can be validated. Figure 7.5 illustrates the regions of interest of the MBRs that are used to compute the directional and distance *decisive* relations. In particular it shows that the MBRs produced by DBSCAN(3, 300) are smaller than the ones produced by DBSCAN(2, 300). In fact, this may give an evidence that the smaller MBRs lead to produce a high number of directional and distance *decisive* relations. Again, this can be validated by the examples appeared in Figure 7.5, in which the MBRs of DBSCAN(2, 300) include more MBRs than the MBRs created by DBSCAN(3, 300).

It is also worth mentioning that DBSCAN(11, 370) has the highest reduction rates for the directional and distance relations compared to the other clustering candidates. This is due to the fact that DBSCAN(11, 370) produces smaller and less CCHs than other candidates. However, its reduction rate for the topological relations is slightly less than other candidates (e.g., DBSCAN(3, 300)) since its CCHs enclosure is not as tight as some other candidates.

### 7.2 Indexing Approaches Experiments

In this section, we first present experiments to evaluate the ability of QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> to reduce the size of graph database ( $\mathcal{G}_D$ ) (see Section 7.2.2).

Next, we present experiments and evaluate the efficiency and performance of the seven matching approaches: (1) Qualitative Layer Matcher (QLM), (2) DBSCAN Matcher (DM), (3) Hybrid Interpretation Tree and B<sup>+</sup>-Tree (HITBT), (4) Qualitative Hash Table Indexing (QHTI), (5) Qualitative Hash Table Compression (QHTC), (6) QHTC of Qualitative Models (QHTC<sup>M</sup>), and (7) QHTC of Object Pairs (QHTC<sup>P</sup>) (see Section 7.2.3, 7.2.4, and 7.2.5). We begin by giving the experimental settings and presenting the experimental results and findings afterwards.

#### 7.2.1 The Experimental Settings of Indexing Approaches

Similar to Section 7.1 and with the same motivation, we used Bremen OSM dataset that contains 8756 labelled objects to evaluate our approaches.

Based on the dataset, three kinds of graph databases have been constructed. The first graph database is  $\mathcal{G}_{\mathcal{D}}$  and has been constructed by computing  $((76.66 * 10^6) - 8756)$  tuples  $((8756 * 8756) - 8756)$ . Each tuple represents the pairs of objects and their spatial relations that are held among them. In our experiments, three types of qualitative spatial relations have been abstracted: topology, direction, and distance. To abstract the topological relations, we used the Dimensionally Extended 9-Intersection Model (DE-9IM) (Clementini et al., 1993) that distinguishes eight topological relations: *equal*, *disjoint*, *meets*, *overlaps*, *contains*, *covers*, *inside*, and *coveredBy*. We applied the Cardinal Direction Model (Skiadopoulos and Koubarakis, 2004) for extended objects to abstract the (single and multi-tile) directional relations: *South*, *SouthWest*, *West*, *NorthWest*, *North*, *NorthEast*, *East*, *SouthEast*, and *Equal*. We abstracted the aspect of distance to qualitative spatial relations using the partition scheme proposed in Section 4.2 to assign one of the four-distance relations: *ZeroDist*, *near*, *medium*, or *far* to each pair of objects. Constructing  $\mathcal{G}_{\mathcal{D}}$  took 5 hours and 48 minutes.  $\mathcal{G}_{\mathcal{D}}$  represents the graph database of QLM. Additionally,  $\mathcal{G}_{\mathcal{D}}$  must also be used by QHTI, QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> to construct their database trees and indices that are required to evaluate these approaches. Table 7.6 lists the constructed trees of the corresponding approaches as well as their index construction time. From the table, it is apparent that HITBT takes less time than other approaches to construct the index. The other approaches needed to sequentially traverse all the tuples of  $\mathcal{G}_{\mathcal{D}}$  and perform merge operation on the related attributes of the tuples.

The second and third graph databases are the  $C_R$  and  $\mathcal{G}_{\mathcal{D}}^C$  (cf. Section 4.4.1.3) respectively that are essential to evaluate DM. Based on the DBSCAN(11, 370) results obtained from Section 7.1, the  $C_R$  is constructed by computing the *decisive* and the *indecisive* relations between the cluster pairs. In addition, the  $\mathcal{G}_{\mathcal{D}}^C$  was constructed by abstracting the spatial relations within the clusters and between the cluster pairs that have the *indecisive* relations. Moreover, three types of spatial relations (topology, direction, and distance) were involved in the abstraction process of the  $C_R$  and  $\mathcal{G}_{\mathcal{D}}^C$ . Table 7.7 outlines the number of abstracted relations of the  $C_R$  and  $\mathcal{G}_{\mathcal{D}}^C$ . In turn, constructing the  $C_R$  and  $\mathcal{G}_{\mathcal{D}}^C$  took 3 hours and 32 minutes which is 27.7% less time needed to construct  $\mathcal{G}_{\mathcal{D}}$ . This occurs since constructing  $C_R$  and  $\mathcal{G}_{\mathcal{D}}^C$  requires computing less spatial relations than the computed spatial relations of  $\mathcal{G}_{\mathcal{D}}$ .

## 7. EMPIRICAL EVALUATION

---

**Table 7.6:** The first level of the constructed trees and their index construction time.

Approach	Constructed Tree(s)	Index Construction [second]
HITBT	$T^B$	2157
$\text{QHTC}^P$	$T^P.SpRel, T^P.OPairs$	7539
$\text{QHTC}^M$	$T^M.SpRel, T^M.OPairs$	7677
QHTI	$T$	8215
QHTC	$T^C$	8939

**Table 7.7:** The number of abstracted relations of the  $C_R$  and  $\mathcal{G}_{\mathcal{D}}^C$ .

	Topology	Direction	Distance
$C_R$	80698	68110	61646
$\mathcal{G}_{\mathcal{D}}^C$	11141640	45533878	64216032
Total	11222338	45601988	64277678

The experiments were performed on a Windows7 platform, 2.1 GHz processor with 4 GB of RAM. Based on the response time of the queries, we examined the performance of our proposed approaches under various parameters.

In each experiment, we varied one parameter, whereas other parameters were kept at their default values. The average response time for answering spatial queries was employed as a major performance measure. In all experiments, we ran the spatial queries five times and we took the average execution time afterwards. Table 7.8 illustrates parameters and their default values. In particular, it shows that the first parameter represents varying the number of queries from 1 to 100 by an increment of 1. In particular, the 100 non-empty results queries were randomly generated by using the `Auto_Queries_Generator`( $\mathcal{G}_{\mathcal{D}}, T^C, 100, 10000, \text{True}, 100$ ) function presented in Section 6.7. The function generates the 100 unique (parameters 5 and 6) queries that retrieve the number of results between 100 and 10000 (parameters 3 and 4). The generation of the random queries aims to examine as many different cases as possible. Furthermore, each query exactly contains a single object pair and three types of spatial relations. For example, Figure 7.6 shows an SQL query which contains three types of spatial relations and one pair of objects. In the second parameter, we similarly varied the number of object pairs in the queries. We extracted the first 10 queries out of the original 100 single pair queries. Afterwards, we iterated over each query and extended it by a new single pair query. Consequently, 10 double object pairs queries were randomly generated,



## 7.2 Indexing Approaches Experiments

**Table 7.8:** Parameter settings.

parameter	setting	Default
Number of spatial queries $\#q$	1, 2, 3, ..., 100	1
Number of object pairs $\#p$	1, 2	1
Number of objects $\#o$ (Cardinality)	1000, 3000, 5000, 8756	8756

```

SELECT *
FROM A
WHERE Reference_Obj='Building' AND Primary_Obj='School'
AND Distance='Near' AND Topology='Disjoint' AND Direction='NorthEast'

```

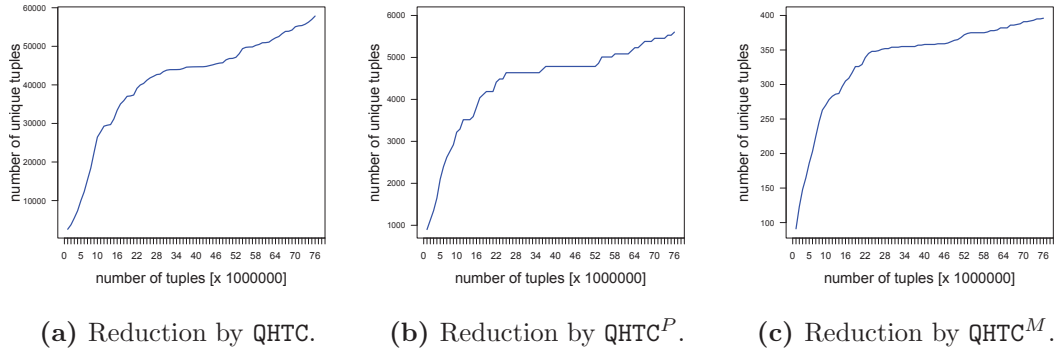
**Figure 7.6:** SQL code: a single pair query with its three spatial relations.

where each pair had three kinds of spatial relations. In the last parameter, we varied the number of objects of the database (the cardinality). We started with 1000 objects. Next, we extended the 1000 objects to 3000 objects, then to 5000 objects, and eventually to 8756 objects.

### 7.2.2 Qualitative Data Reduction

We evaluate the reduction qualitative data capability of the QHTC,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$  compared to the original size of the graph database ( $\mathcal{G}_{\mathcal{D}}$ ). Simplified, we denote the size of  $\mathcal{G}_{\mathcal{D}}$  by  $|\mathcal{G}_{\mathcal{D}}|$ .

Figure 7.7 conveys that all three indexing approaches reduce considerably the tuples of object pairs, the spatial relations, and a combination of them compared to  $|\mathcal{G}_{\mathcal{D}}|$ .



**Figure 7.7:** The space reduction rates of  $\mathcal{G}_{\mathcal{D}}$  by QHTC,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$ .

## 7. EMPIRICAL EVALUATION

---

Figure 7.7(a) illustrates that the QHTC reduces the size of  $\mathcal{G}_{\mathcal{D}}$  to  $T^C$  that acts as 58024 unique tuples instead of  $76.66 * 10^6$  tuples. This indicates that the size of  $T^C$  of QHTC is approximately  $75.6 * 10^{-3} \%$  compared to  $|\mathcal{G}_{\mathcal{D}}|$ . The reduction rate of the tuples of the Object Pairs ( $OP$ ) of  $\mathcal{G}_{\mathcal{D}}$  is shown in Figure 7.7(b), in which  $\text{QHTC}^P$  reduces the recurrences of  $OP$  tuples to be  $T^P.OPairs$  that represents 5603 unique tuples instead of  $76.66 * 10^6$  tuples. Accordingly, the size of  $T^P.OPairs$  is approximately  $73.1 * 10^{-4} \%$  compared to the size of  $OP$ . Finally, Figure 7.7(c) depicts that  $\text{QHTC}^M$  significantly reduces the number tuples of the Spatial Relations ( $SR$ ) with respect to  $|\mathcal{G}_{\mathcal{D}}|$ . In particular,  $\text{QHTC}^M$  reduces the repeated  $SR$  tuples of  $\mathcal{G}_{\mathcal{D}}$  to  $T^M.SpRel$  that contains 397 unique tuples instead of  $76.66 * 10^6$  tuples. The size of  $T^M.SpRel$  is therefore approximately  $51.8 * 10^{-5} \%$  compared to the size of  $SR$  of  $\mathcal{G}_{\mathcal{D}}$ .

### Discussion:

The previously mentioned approaches showed a high capability of reducing the data of  $\mathcal{G}_{\mathcal{D}}$ . Nevertheless, the reduction rates are variant with respect to the number of tuples of  $\mathcal{G}_{\mathcal{D}}$ . For example, in a case of the  $T^C$  of QHTC (Figure 7.7(a)) 26448 unique tuples are detected (reduced to 26448 tuples) in the first 10 million tuples, while 10582 unique tuples are found in the next 10 millions tuples. This occurs because the number of repeated tuples of  $\mathcal{G}_{\mathcal{D}}$  is not equally distributed with the number of tuples of  $\mathcal{G}_{\mathcal{D}}$ . The reduction rates by  $\text{QHTC}^P$  and  $\text{QHTC}^M$  are variant with the number of  $\mathcal{G}_{\mathcal{D}}$  tuples as well. Similar to QHTC, these variants of reduction rates can be justified. However, the reduction rates by  $\text{QHTC}^P$  and  $\text{QHTC}^M$  are very high, since the number of labels of spatial relations or object pairs is limited. For example, regarding  $\text{QHTC}^M$  the 9-intersection model contains only eight spatial relations and therefore eight labels.

### 7.2.3 Varying the Number of Queries

Here we evaluate the scalability of the matching approaches by varying the number of queries ( $\#q$ ) on the complete database ( $\#o = 8756$ ). We accumulate the execution time from a single query to the maximum of 100 queries, i.e. the execution for  $n \geq 1$  queries is the average execution time of the  $n^{th}$  query plus the aggregated execution time of the previous  $n - 1$  queries.

To begin, the **naive approach** presented in Section 4.3 does not scale at all. For example, the average execution time of each query (single object pair) is approximately

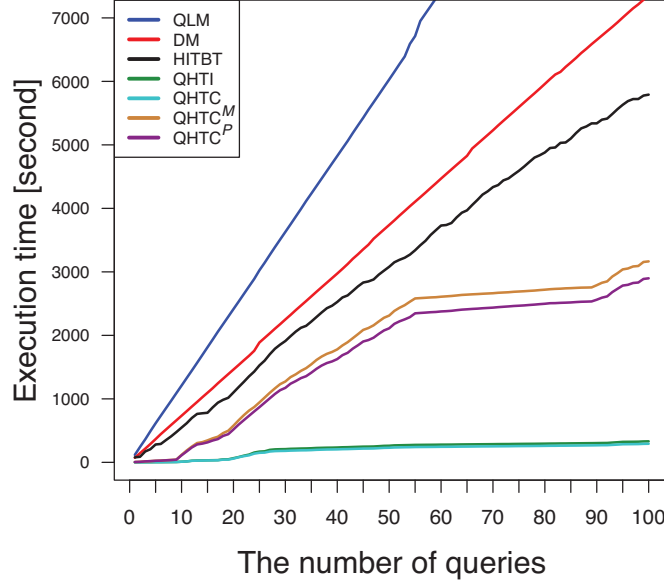


Figure 7.8: Varying the number of single pair queries.

16867 seconds (approx. 4 hours and 41 minutes), which is in fact greater than the execution time of the 50 queries of the rest approaches together. Therefore, we focus our attention on comparing other approaches than the **naive approach**. We depict the results in Figure 7.8 and Table 7.9. Whereas processing time for the first query for QHTI is  $\approx 1.3$  seconds, 1.05 seconds for QHTC respectively, QHTC<sup>M</sup> needs  $\approx 8.91$  seconds, and QHTC<sup>P</sup>  $\approx 8.45$  seconds. Eventually, QLM needs  $\approx 242.51$  seconds, DM  $\approx 144.33$  seconds, and HITBT  $\approx 87.71$  seconds.

The overall response time for 100 queries for QHTI and QHTC is not that much different:  $\approx 331.08$  seconds for QHTI and  $\approx 292.83$  seconds for QHTC. In turn, QHTC<sup>P</sup> and QHTC<sup>M</sup> require  $\approx 2898$  and  $\approx 3165$  seconds respectively to process all queries. To process all the queries, QLM needs  $\approx 12072$  seconds, DM  $\approx 7346$  seconds, and HITBT  $\approx 5791$  seconds. Therefore, QHTC performs best on average for a single query with  $\approx 2.93$  seconds ( $\sigma = 4.62$ ), followed by QHTI with  $\approx 3.31$  seconds ( $\sigma = 5.41$ ).

QHTC<sup>P</sup> and QHTC<sup>M</sup> run approximately 10 and 11 times slower than QHTC respectively, QLM, DM, and HITBT 40, 25, and 19 times slower than QHTC respectively. Additionally, QHTI and QHTC progress in a very similar manner with little differences only, i.e., similar

## 7. EMPIRICAL EVALUATION

---

**Table 7.9:** The minimum, maximum, average, and standard deviation ( $\sigma$ ) execution time measured by seconds for the spatial queries.

Approach	Minimum Exec. Time	Maximum Exec. Time	Average Exec. Time	$\sigma$
QLM	81.14	237.31	120.73	20.66
DM	41.10	130.99	73.46	11.25
HITBT	5.63	101.16	57.97	24.41
QHTC <sup>M</sup>	1.42	85.25	31.65	27.24
QHTC <sup>P</sup>	1.27	77.17	28.98	25.17
QHTI	0.25	32.55	3.31	5.41
QHTC	0.22	26.67	2.93	4.62

min, max, and average values. Also standard deviation ( $\sigma$ ) in processing time is low in both cases.

The results of Table 7.9 support and agree with the results presented previously. The performance and scalability of our approaches stay relatively the same in terms of the minimum, maximum, and average values of the queries. For example, QHTI and QHTC have the lowest min, max, and average values. Oppositely, QLM and DM have the highest min, max, and average values.

### Discussion:

Based on the results described above, QHTI and QHTC clearly outperform the other approaches, when 100 unique queries are processed. Figure 7.8 also exhibits that QHTC<sup>M</sup> and QHTC<sup>P</sup> surpass QLM and HITBT respectively. In turn, HITBT outperforms QLM and DM which in turn exceeds QLM.

QHTI and QHTC scaled very similar when compared to the other approaches. That is, in contrast to other approaches, QHTI and QHTC need to visit  $\mathcal{G}_{\mathcal{D}}$  only one time, since the attributes of  $\mathcal{G}_{\mathcal{D}}$  are merged (cf. Section 5.2 and 5.3). However, QHTC scales slightly better than QHTI since QHTI needs to traverse the tuples (usually many and repeated tuples) with same hash values of the queries (cf. Section 5.2).

The next two best approaches are QHTC<sup>P</sup> and QHTC<sup>M</sup> respectively. They perform particularly well due to the fact that the matching of queries is done by using the two-stage hierarchy. The two-stage hierarchical approach uses the labels of spatial relations (in QHTC<sup>M</sup>) or object pairs (in QHTC<sup>P</sup>) to range the matches, then the ranged ones are searched (cf. Section 5.4 and 5.5).

$\text{QHTC}^P$  answers the queries faster than  $\text{QHTC}^M$ . This gives an indication that the labels of object pairs ( $T^P.OPairs$ ) range the matchings more efficient than the spatial relations labels ( $T^M.SpRel$ ) of  $\text{QHTC}^M$ . The performance of  $\text{QHTC}^M$  and  $\text{QHTC}^P$  depends on the tuple recurrences and the distribution of these recurrences. In other words,  $\text{QHTC}^M$  might even surpass  $\text{QHTC}^P$  in some other cases.

In order to answer the queries, HITBT exhibits a better performance than DM. HITBT takes advantage of the  $B^+$ -tree index that accelerates the search process (cf. Section 5.1). On the contrary, DM is required to perform a self join to obtain the objects of clusters that are in a *decisive* relation (cf. Section 4.4.1.4). In addition, DM needs to apply another join to retrieve the results of the clusters.

DM scales better than QLM due to the fact that it needs to visit a smaller number of tuples than QLM.

QLM does not scale well compared to other approaches for two reasons: (1) it needs to visit the attributes of  $\mathcal{G}_D$  several times and to perform joins and (2) no-indexing is provided which implies an arbitrary and exhaustive search.

In turn, the **naive approach** does not scale at all, since the spatial relations need to be computed at run time which is a very expensive process. Additionally, joins must be done twice, one for computing the spatial relations, and another one for matching the queries.

Finally, we note that the response times of our approaches are greatly increased to process the queries from 10 to 100. This is due to the fact that the queries may cover high density zones (wrt. objects) and they retrieve many results from databases.

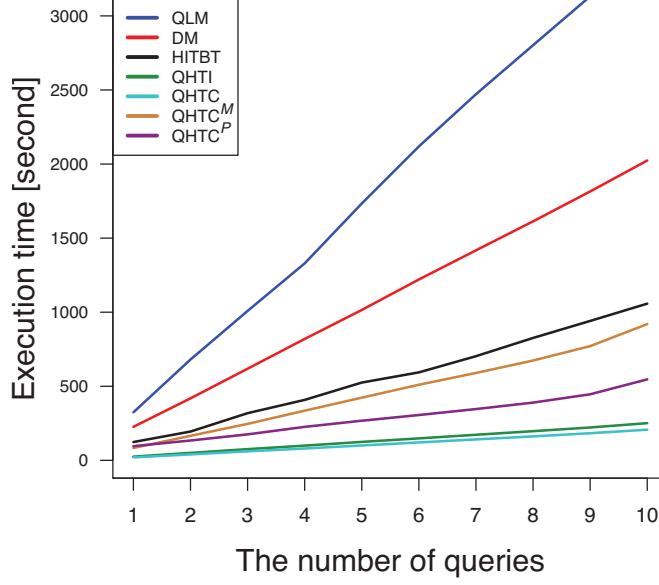
### 7.2.4 Varying the Number of Pairs

In this sub-section we vary the number of pairs ( $\#p$ ) to examine the performance of the proposed approaches ( $\#p \in \{1, 2\}$ ). As depicted in Figure 7.10 varying  $\#p$  from 1 to 2 has a significant impact on the response time to process the 10 spatial queries by our approaches.

In general, Figure 7.10 illustrates that processing the two object pair queries ( $\#p=2$ ) requires more time than the single object pairs ones ( $\#p=1$ ). Figure 7.10(a) conveys that QLM is able to answer the all queries in less than 1209 seconds when  $\#p = 1$  and in less than 3466 seconds if  $\#p = 2$ . Compared to the spatial queries when  $\#p = 1$ , QLM requires approximately triple response time when  $\#p = 2$ .

## 7. EMPIRICAL EVALUATION

---



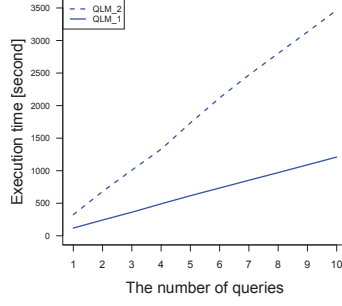
**Figure 7.9:** Varying the number of double pairs queries.

DM and HITBT answer the spatial queries when  $\#p = 1$  in less than 734 and 547 seconds respectively, and in less than 2023 and 1057 seconds respectively when  $\#p = 2$  (Figure 7.10(b) and 7.10(c)). On average and compared to spatial queries when  $\#p = 1$ , DM and HITBT have a little less than triple and double response time respectively, for processing spatial queries when  $\#p = 2$ .

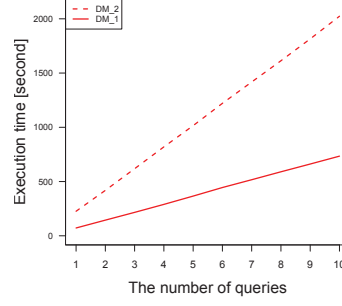
In order to process all spatial queries with  $\#p = 2$ , QHTC<sup>P</sup> takes 546 seconds while QHTC<sup>M</sup> takes 919 seconds (Figure 7.10(e) and 7.10(d)). Thus, when  $\#p = 2$  the response time of all the queries processed by QHTC<sup>M</sup> and QHTC<sup>P</sup> increases by a factor of approximately 9 and 5 respectively in comparison to when  $\#p = 1$ .

Finally, QHTI and QHTC are able to answer the spatial queries in less than approximately 251 and 206 seconds respectively (Figure 7.10(f) and 7.10(g)) when  $\#p = 2$ . Furthermore, compared to the spatial queries with  $\#p = 1$ , the processing time of all the spatial queries processed by QHTI and QHTC with  $\#p = 2$  is increased by a factor of approximately 24 and 20 on average respectively. Despite the significant difference between the response time for spatial queries when  $\#p = 1$  and when  $\#p = 2$ , the performance and scalability of our approaches stay relatively the same as those presented

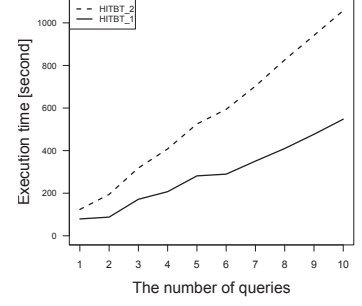
## 7.2 Indexing Approaches Experiments



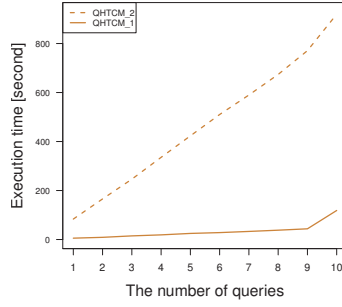
(a) Comparing QLM



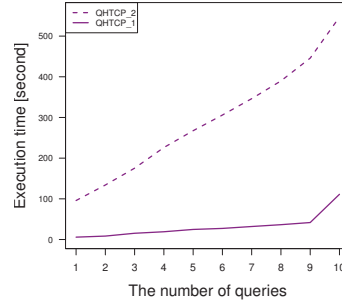
(b) Comparing DM



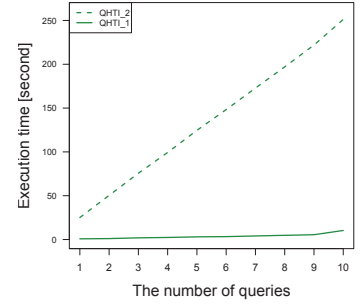
(c) Comparing HITBT



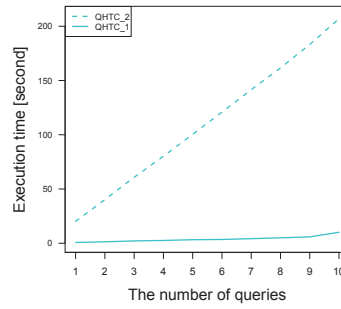
(d) Comparing  $\text{QHTC}^M$



(e) Comparing  $\text{QHTC}^P$



(f) Comparing QHTI



(g) Comparing QHTC

**Figure 7.10:** Comparing our approaches by varying the number of object pairs.

## 7. EMPIRICAL EVALUATION

---

in Section 7.2.3 (see Figure 7.9). Figure 7.9 shows that QHTI and QHTC are more scalable than the other approaches. In contrast, QLM and the DM do not scale well in comparison to the other approaches.

### Discussion:

To begin, varying the number of pairs ( $\#p$ ) has a large increase in the response time by our approaches to process the spatial queries. In particular, the response time of QLM, DM, and HITBT rose up when  $\#p = 2$ . By increasing the number of pairs, it is required to increase the number of (self) join operations as well.

Comparatively, peaking the response time of  $\text{QHTC}^M$  and  $\text{QHTC}^P$  when  $\#p = 2$  can be justified. However, the response time of QHTC and QHTI is dramatically increased when  $\#p = 2$  due to two reasons. First, many of the tuples need to be processed by QHTC and QHTI in order to answer the queries. Second, the number of the retrieved results is considerably higher than the ones when  $\#p = 1$ .

### 7.2.5 Varying the Number of Objects

In this sub-section, we consider a single query ( $\#q = 1$ ) regarding different sizes of the underlying database ( $\#o = 1000, \dots, 8756$ ). Based on the processing time behavior we extracted four main classes of queries (cf. Figure 7.11). We gave a prototypical query for each class and according results in Table 7.10. Although, most other cases behave very similar to one of these cases, some queries may be considered as mixed cases of these four.

From Figure 7.11, it is apparent that QHTI and QHTC surpass the other approaches when processing the four queries. Oppositely, QLM has the worst performance to process the queries. In particular, Figure 7.11(a) shows the result of the first query, in which  $\text{QHTC}^M$  and  $\text{QHTC}^P$  outperform HITBT. Furthermore, HITBT surpasses DM and QLM. It also indicates that the response times of all the approaches rose sharply when the number of objects  $\#o > 5000$ . In fact, according to the Table 7.10 the number of retrieved results increases sharply when  $\#o > 5000$ . In addition,  $\text{QHTC}^M$  and  $\text{QHTC}^P$  take similar processing time as HITBT when  $\#o \leq 5000$ . Figure 7.11(b) and 7.11(c) illustrate the results of the second and third query, which are very similar to the results of the first query. In these queries, HITBT clearly exceeds DM when  $\#o$  is varied from 1000 to 8756.



## 7.2 Indexing Approaches Experiments

**Table 7.10:** Four spatial queries and their cardinalities as well as their numbers of the retrieved results.

Query: Primary Obj. {Spatial Rels} Reference Obj.	Card. (N)	# of retrieved results
Building {Medium-Disjoint-East} Park	N=1000	11
	N=3000	93
	N=5000	728
	N=8756	1768
Building {Near-Disjoint-East} Park	N=1000	9
	N=3000	501
	N=5000	591
	N=8756	920
Pitch {Near-Disjoint-NorthWest:West} Building	N=1000	50
	N=3000	751
	N=5000	756
	N=8756	1738
Building {Near-Disjoint-NorthEast} School	N=1000	3
	N=3000	1044
	N=5000	1065
	N=8756	1289

The results of the fourth query can be seen in Figure 7.11(d). Interestingly,  $\text{QHTC}^M$  outperforms  $\text{QHTC}^P$  when  $\#o < 5000$  and DM surpasses HITBT when  $\#o \leq 5000$ .

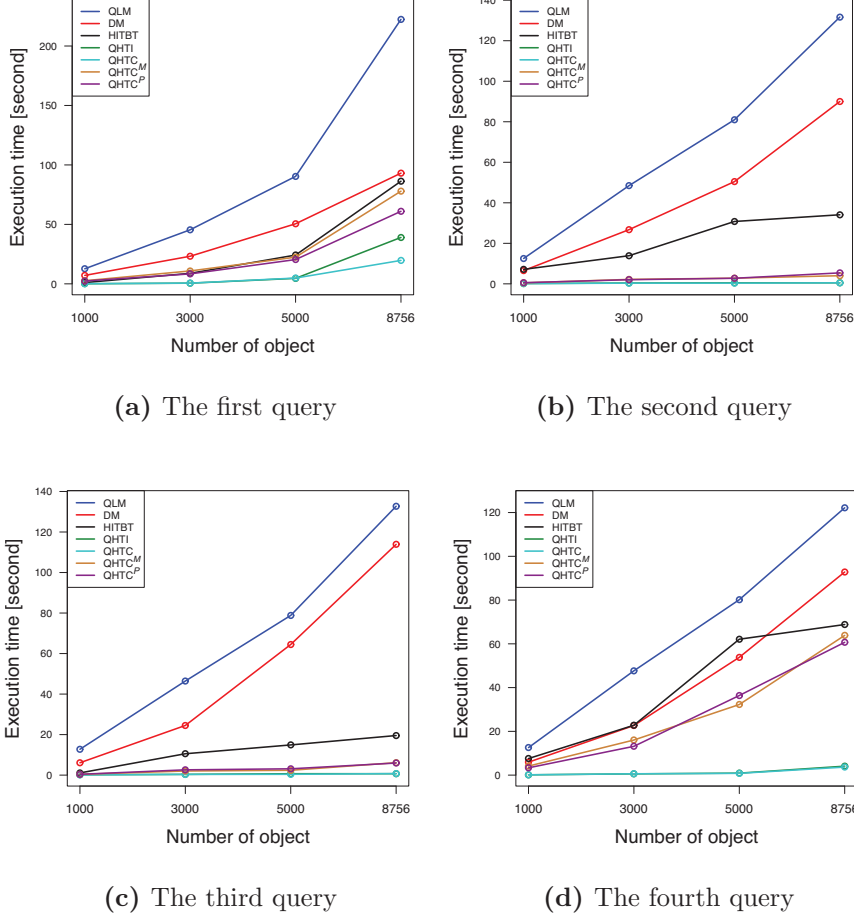
### Discussion:

Broadly speaking, changing the number of objects (cardinalities) has a considerable impact on the performance and the behavior of the presented approaches in order to process the four queries.

QLM does not scale well due to the fact that the database tuples must be searched arbitrarily without any order. Furthermore, the joins are required to retrieve the corresponding results. Conversely, no joins are needed in the case of QHTI and QHTC which justifies their ability to scale better than the other approaches.

The response time of our approaches when  $\#o < 5000$ , are significantly less than the ones when  $\#o > 5000$ . This is most likely due to the fact that they only need to

## 7. EMPIRICAL EVALUATION



**Figure 7.11:** Comparison: varying the number of objects of the database (cardinality).

retrieve a small number of objects if  $\#o < 5000$  (see Table 7.10) compared to other cardinalities ( $\#o > 5000$ ).

Regarding the first, second, and third queries, the performance of our approaches can be generally validated similarly to the descriptions and justifications mentioned in Section 7.2.3 and 7.2.4. Nevertheless, Figure 7.11(a) represents interesting results for the first query, in which HITBT,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$  perform and scale similarly when  $\#o \leq 5000$ . That is, the ranging strategies used by  $\text{QHTC}^M$  and  $\text{QHTC}^P$  were not efficient enough to range either the spatial relations or the object pairs.

Figure 7.11(d) also shows interesting results for the fourth query. In the sense that DM surpasses HITBT when  $\#o \leq 5000$  and  $\text{QHTC}^M$  outperforms  $\text{QHTC}^P$  when  $\#o = 5000$ .

Regarding DM and HITBT, our explanation is that the latter has to partially perform an exhaustive search, due to the fact that the dataset with  $\#o = 5000$  may contain most of the labels of spatial relations and object pairs. In other words, HITBT may have to traverse many branches of the levels of B<sup>+</sup>-tree index (for all or some database attributes) to extract the nodes with same labels of the query. This argument can be complementarily supported by the results shown in the Table 7.10 that conveys that the most results are retrieved from database when  $\#o = 5000$ .

Finally, it is noticeable that the response time varies from query to another, since the queries may cover various density zones (wrt. objects) which may make the response times of the queries vary as well.

## 7.3 Synthetic Data Evaluation

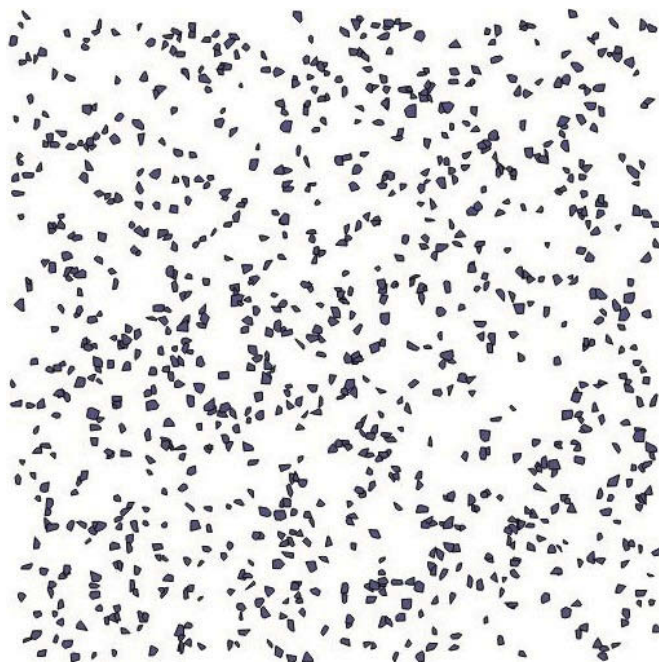
In this section we evaluate our approaches by using a synthetic data to examine their performance. Firstly, we report about the reduction rates achieved by the DBSCAN approach (see Section 7.3.1). Afterwards, we evaluate the capability of qualitative data reduction by QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> (see Section 7.3.2.1). In Section 7.3.2.2, we examine the scalability of our matching approaches based on the execution time of spatial queries.

### 7.3.1 Clustering Experiments on a Synthetic Data

In our experiments, we have used a synthetic dataset that has been stored in the PostgreSQL/PostGIS database and contained 1000 polygons. The 1000 polygons have been randomly generated and uniformly distributed in  $D \times D = 500 \times 500$  fixed-size workspace. Furthermore, each polygon is randomly assigned a label (e.g., park). Moreover, all dataset objects have the same two-dimension extents. Figure 7.12 demonstrates a snapshot the randomly generated polygons of the synthetic dataset. We justify the choice of data distributions in terms of our synthetic dataset. In contrast to the real-world dataset we have used in Section 7.1 and 7.2, we here use uniform distributed dataset for two reasons: (1) to show that our approaches are scalable and efficient even when we use a different dataset and (2) regarding querying and clustering, the objects in real-world datasets seem to be more efficient to handle than the objects in uniform distributed dataset.

## 7. EMPIRICAL EVALUATION

---



**Figure 7.12:** A snapshot of a synthetic dataset.

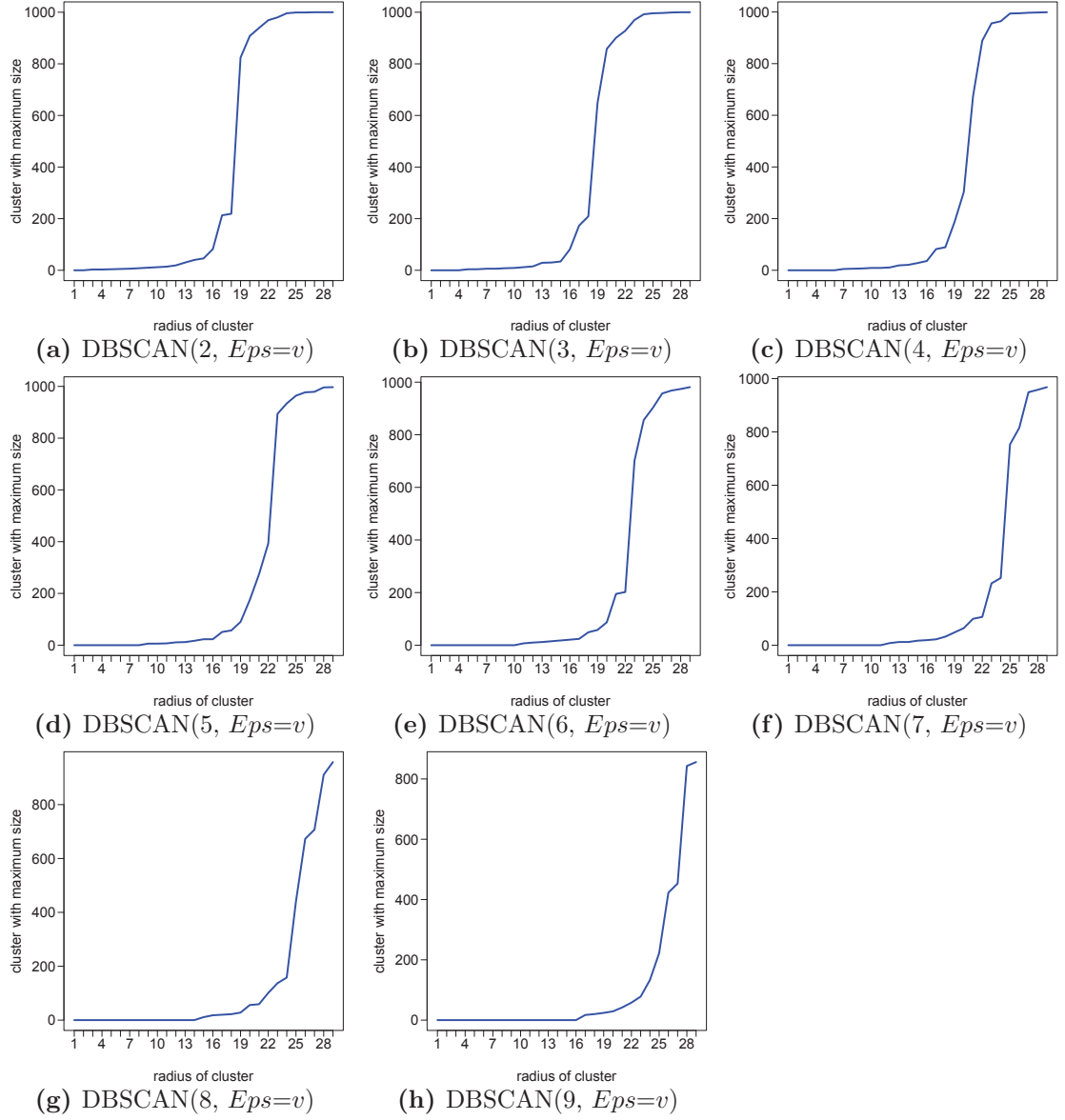
Now, the reduction process by DBSCAN happens in two stages: (1) Filtering Clustering Candidates and (2) Selecting Clustering Candidate.

### (1) Filtering Clustering Candidates:

In order to select clustering candidates, we vary the *MinPts* and the *Eps* values of DBSCAN. DBSCAN terminates the clustering process when  $\geq 50\%$  of the database objects are clustered and the maximum size cluster is at least doubled.

We start by setting the *MinPts*=2 and we iterate over all the *Eps* values from 1 to 30 incremented by 1. Similarly, we iterate over all the *MinPts* values from 3 to 9 increased by 1 (see Figure 7.13). From Figure 7.13, it is apparent that peaking situations occur when  $Eps \geq 15$ . Based on the experiments appeared in Figure 7.13, we select the eight clustering candidates. Table 7.12 illustrates their *MinPts*, *Eps*, their number of clusters, their maximum cluster size, and their number of outliers. In all the experiments, DBSCAN took between 3 and 7.5 seconds to cluster the objects.

Similar to Section 7.1.2, we compute Pearson correlation coefficient ( $r$ ) and  $p$ -value to measure the strength and significance of the relationship between pairs of variables. As pairs of variables of interest, Table 7.11 shows the values of  $r$  and  $p$ -value between



**Figure 7.13:** Snapshots of  $DBSCAN(MinPts, Eps=v)$ ,  $v$ : the radius of clusters varied from 1 to 30 degrees incremented by 1.

## 7. EMPIRICAL EVALUATION

---

**Table 7.11:**  $r$  and  $p$ -value for  $MinPts$  with other variables.

Measure		$Eps$	# of clusters	# of outliers
$r$	$MinPts$	0.877	-0.979	0.612
$p$ -value	$MinPts$	0.004	0.0000991	0.106

**Table 7.12:** The clustering candidates of DBSCAN experiments.

$MinPts$	$Eps$	# of clusters	Maximum cluster size	# of outliers
2	15	87	46	113
3	15	73	34	241
4	16	57	36	327
5	18	45	57	276
6	19	40	58	309
7	22	27	106	211
8	21	26	59	423
9	21	17	133	301

the  $MinPts$  and other variables.

The correlation coefficient between the  $MinPts$  and the  $Eps$  is significant ( $p$ -value  $< 0.05$ ), *positive*, “very strong”, and linear. The correlation between the  $MinPts$  and the number of clusters is significant, *negative*, and “very strong”. Lastly, the correlation between the  $MinPts$  and the number of outliers is *not* significant ( $p$ -value  $> 0.05$ ), *positive*, and “strong”.

### (2) Selecting Clustering Candidate:

In order to select a clustering candidate, we compare the candidates by calculating the number of spatial relations per qualitative aspect that can be saved by each clustering candidate. The reduction rates of the topological, directional, and distance relations as well as their average reduction are reported in Table 7.13.

### Discussion:

Among the other candidates, the first candidate DBSCAN(2, 15) offers the highest directional, distance, and average reduction rates with 79.71 %, 89.60 %, and 86.83 % respectively. That is, the DBSCAN(2, 15) candidate generates smaller CCHs and MBRs than those produced by others and has the lowest number of outliers.

**Table 7.13:** The reduction rates for the topological, directional, and distance relations as well as their average reduction.

<i>MinPts</i>	<i>Eps</i>	Topology Red.	Direction Red.	Distance Red.	Avg Red.
<b>2</b>	15	91.18 %	79.71 %	89.60 %	86.83 %
<b>3</b>	15	92.52 %	67.90 %	88.47 %	82.96 %
<b>4</b>	16	87.18 %	64.83 %	83.72 %	78.58 %
<b>5</b>	18	89.44 %	59.86 %	84.26 %	77.85 %
<b>6</b>	19	61.60 %	50.21 %	59.31 %	57.04 %
<b>7</b>	22	90.58 %	41.69 %	77.33 %	69.87 %
<b>8</b>	21	78.350 %	51.13 %	73.97 %	67.82 %
<b>9</b>	21	70.94 %	51.52 %	68.08 %	63.52 %

Accordingly, the number of the directional and distance *decisive* relations are increased which leads to increase the reduction rate on average.

The next candidate DBSCAN(3, 15) provides the highest topological reduction rate with 92.52 %. This topological reduction rate is achieved by the candidate, as the size of some generated clusters are slightly increased in comparison to ones provided by DBSCAN(2, 15) and no other clusters are included in the increased ones. In addition, the shapes of clusters generated by DBSCAN(3, 15) are completely changed compared to the ones generated by DBSCAN(2, 15). Consequently, the DBSCAN(3, 15) candidate was able to enclose its clusters using the CCHs more tightly than DBSCAN(2, 15) which leads to reduce the number of the non-*disjoint* relations and thus increase the reduction rate. However, the penalty of such increase of the clusters size is that the number of the directional and distance *decisive* relations are decreased (cf. Section 7.1.3). This led to reduce the amounts of the directional and distance reduction rates.

### 7.3.2 Indexing Approaches Experiments on a Synthetic Data

In this sub-section, we present two kinds of experiments: (1) evaluating the ability of QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> to reduce the size of  $\mathcal{G}_{\mathcal{D}}$  (Section 7.3.2.1) and (2) evaluating the scalability of our approaches (Section 7.3.2.2).

Similar to Section 7.3.1, we have used the synthetic dataset that contains 1000 labelled polygons to evaluate our approaches. Based on the dataset, three kinds of graph databases have been constructed. The first graph database is  $\mathcal{G}_{\mathcal{D}}$  and has been constructed by computing 999000 tuples ((1000\*1000)-1000). Each tuple represents the

## 7. EMPIRICAL EVALUATION

---

**Table 7.14:** The first level of the constructed trees and their index construction time.

Approach	Constructed Tree(s)	Index Construction [second]
HITBT	$T^B$	34
$\text{QHTC}^P$	$T^P.SpRel, T^P.OPairs$	391
$\text{QHTC}^M$	$T^M.SpRel, T^M.OPairs$	417
QHTI	$T$	440
QHTC	$T^C$	501

pairs of objects and their spatial relations that are held among them. In our experiments, three types of qualitative relations have been abstracted: topology, direction, and distance. Constructing  $\mathcal{G}_D$  took 229 seconds.  $\mathcal{G}_D$  represents the graph database of QLM. Additionally,  $\mathcal{G}_D$  must also be used by QHTI, QHTC,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$  to construct their database trees and indices that are essential to evaluate these approaches. Table 7.14 lists the constructed trees of the corresponding approaches as well as their index construction time.

The second and third graph databases are the  $C_R$  and  $\mathcal{G}_D^C$ , respectively. They are required to evaluate the DM approach. Constructing the  $C_R$  and  $\mathcal{G}_D^C$  took 36 seconds. The  $C_R$  is constructed using the DBSCAN(2, 15) results gathered from Section 7.3.1.

The experiments were performed on a Windows7 platform, 2.1 GHz processor with 4 GB of RAM. The average response time for answering spatial queries is employed as a major performance measure. In the experiments, we ran the spatial queries five times and we took the average execution time afterwards. In particular, we varied the number of queries from 1 to 50 incremented by 1. The 50 non-empty results queries were randomly generated by using the `Auto_Queries_Generator( $\mathcal{G}_D, T^C, 100, 10000, \text{True}, 50$ )` function presented in Section 6.7. The function generated the 50 unique (parameters 5 and 6) queries that retrieved the number of results between 100 and 10000 (parameters 3 and 4). Furthermore, each query contained exactly a single object pair and three types of spatial relations.

### 7.3.2.1 Qualitative Data Reduction

We evaluate the reduction qualitative data capability of the QHTC,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$  compared to the original size of the graph database ( $\mathcal{G}_D$ ). The size of  $\mathcal{G}_D$  is 999000 and denoted by  $|\mathcal{G}_D|$ .



### 7.3 Synthetic Data Evaluation

**Table 7.15:** The number of detected unique tuples of QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> as well as their new and reduced graph size.

Approach	Number of unique tuples	Reduced graph size %
QHTC	$T^C=9295$	0.00931 %
QHTC <sup>M</sup>	$T^M.SpRel=140$	0.00014 %
QHTC <sup>P</sup>	$T^P.OPairs=1662$	0.001664 %

The reduction rates of the QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> in comparison to  $|\mathcal{G}_D|$  are listed in Table 7.15. For example, QHTC reduces the recurrences of  $\mathcal{G}_D$  tuples to be 9295 unique tuples, which is approximately 0.00931 % (9295/999000) in comparison to  $|\mathcal{G}_D|$ . The reduction rates of the aforementioned approaches given in Table 7.15 can be validated by using the justifications given in Section 7.2.2.

#### 7.3.2.2 Varying the Number of Queries

This sub-section presents a comparison between the approaches by varying the number of queries ( $\#q$ ) in order to evaluate their scalability.

Similar to Section 7.2.3, the **naive approach** does not scale at all, since the spatial relations need to be computed at run time. Accordingly, we concentrate on the other approaches than the **naive approach**.

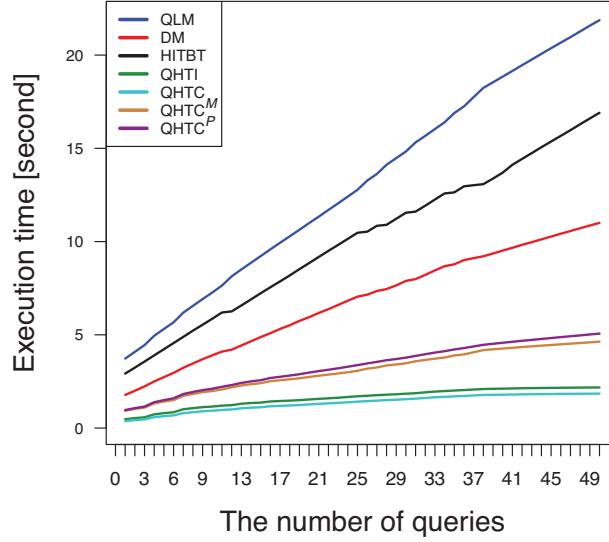
The response time of 50 accumulative spatial queries are illustrated in Figure 7.14. In turn, processing time for the all spatial queries for QHTI is  $\approx 2.2$  seconds, 1.9 seconds for QHTC respectively, QHTC<sup>M</sup> needs  $\approx 4.7$  seconds and QHTC<sup>P</sup>  $\approx 5.1$  seconds. Eventually, QLM needs  $\approx 21.1$  seconds, DM  $\approx 11$  seconds, and HITBT  $\approx 16.91$  seconds. The overall response time for 50 queries for QHTI and QHTC is not that much different. The latter observation holds for QHTC<sup>P</sup> and QHTC<sup>M</sup> as well. QHTC<sup>P</sup> and QHTC<sup>M</sup> run approximately 2.5 and 2.7 times slower than QHTC respectively. In turn, HITBT, DM, and QLM run approximately 9, 6, and 12 times slower than QHTC respectively.

#### Discussion:

The previous experiment showed that QHTC exceeds QHTI that outperforms the other approaches. In contrast to others, QHTI and QHTC need to visit  $\mathcal{G}_D$  only once as the attributes of  $\mathcal{G}_D$  are merged (cf. Section 5.2 and 5.3). Figure 7.14 conveys that QHTC<sup>M</sup> surpasses QHTC<sup>P</sup> that exceeds HITBT, DM, and QLM. In turn, DM outperforms HITBT that

## 7. EMPIRICAL EVALUATION

---



**Figure 7.14:** Varying the number of single pair queries.

surpasses QLM. Unlike the results mentioned in Section 7.2.3, here  $\text{QHTC}^M$  scales better than  $\text{QHTC}^P$ . That means that the labels of the relations of  $\text{QHTC}^M$  tree range the matchings more efficient than the labels of object pairs of  $\text{QHTC}^P$  tree. Another difference in comparison to the results explained in Section 7.2.3 is that DM answers the queries faster than HITBT. Even though DM needed to apply joins to retrieve the results of the clusters, it needed to visit drastically smaller number of tuples than QLM (cf. Section 7.3.1). Moreover, the joins are applied on a small number of objects (1000 objects), which does not make a big influence on the DM performance. The last difference is that the processing time by all the approaches is considerably reduced. Furthermore, some approaches such as QLM showed a good scalability. This may indicate that the QLM might be applicable to cope with small datasets rather than large ones.

## 7.4 Summary

This chapter demonstrated an empirical evaluation. We analyzed and evaluated the ability of the approaches presented in Chapter 4 and 5 to reduce space and time complexity that are related to process spatial queries on the spatial databases.

First, we have carried out two kinds of experiments on a real-world dataset to evaluate performance of our approaches: (1) the space reduction of the graph databases and (2) the execution time of the spatial queries. In Section 7.1 we developed a novel methodology to parametrize DBSCAN. By using the appropriate values for  $Eps$  and  $MinPts$ , DBSCAN showed a good ability to reduce the size of the graph database ( $\mathcal{G}_{\mathcal{D}}$ ) with approximately 47.43 % average reduction rate. Next, in Section 7.2.2 we have evaluated the efficiency of  $\text{QHTC}$ ,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$  to reduce the size of the  $\mathcal{G}_{\mathcal{D}}$ . In particular, the approaches were able to considerably reduce the size of  $\mathcal{G}_{\mathcal{D}}$  by aggregating the exact labels of spatial relations, the pairs of objects, or a combination of them, since there were many recurrences of those labels. In Section 7.2, we examined the efficiency of our approaches to process random spatial queries. In order to examine our approaches, several spatial queries were randomly generated as follows: (i) 100 non-empty results queries for the experiment of varying number of queries, (ii) 10 non-empty results queries for the experiment of varying number of pairs, and (iii) 4 non-empty results queries for the experiment of varying number of objects. The spatial queries were randomly generated to cover as many cases as possible. Regarding the execution time of the spatial queries, in all experiments the scalability of the approaches can be given in the following descending order:

$\text{QHTI} > \text{QHTC} > \text{QHTC}^P > \text{QHTC}^M > \text{DM} > \text{QLM} > \text{the naive approach}$ .

The hash-based approaches showed better scalability than others, due to the fact that the attributes of database table were totally or partially merged. That means that the hash-based approaches needed to visit the database table fewer times than the other approaches.

Similarly in Section 7.3 we have run the aforementioned experiments on a synthetic dataset to examine the behaviour of our approaches as well as their space and time scalability. Regarding the qualitative data reduction, DBSCAN has demonstrated a strong ability to reduce the size of  $\mathcal{G}_{\mathcal{D}}$  with approximately 86.83 % average reduction

## 7. EMPIRICAL EVALUATION

---

rate. This reduction rate when DBSCAN has been applied to the synthetic dataset was extremely higher than the one achieved when it was applied to the real-world dataset.

Next, we have tested the reduction capability of  $\text{QHTC}$ ,  $\text{QHTC}^M$ , and  $\text{QHTC}^P$  to reduce the size of  $\mathcal{G}_{\mathcal{D}}$ , where the approaches have shown an ability to drastically reduce  $\mathcal{G}_{\mathcal{D}}$  size.

Lastly, we have evaluated the response time of spatial queries processed by our approaches. Our experiments suggested that the hash-based approaches were able to answer the queries faster than the other approaches for the reasons mentioned above.

## Chapter 8

# Conclusions

This chapter concludes the dissertation by summarizing our contributions and discussing directions for future work.

## 8.1 Summary

We started this work with observing that most work on geo-spatial databases has been focused on developing novel and powerful techniques to process quantitative spatial queries but not qualitative ones.

However, as we argued that it is more natural and intuitive for humans to query geo-spatial databases by means of qualitative terms than by quantitative values. This type of queries are called Qualitative Spatial Queries (QSQs). Therefore, we have integrated the appropriate qualitative spatial representations into Spatial Data-Base Management Systems (SDBMSs) to allow the qualitative and intuitive formalism of queries in GISs. We have integrated three kinds of qualitative models into the SDBMSs: (1) topology, (2) direction, and (3) distance.

Next, we have used the qualitative models to abstract the three aforementioned kinds of spatial relations and to store them in a Qualitative Spatial Layer (QSL) of spatial databases. This has led to avoid the additional cost of the abstraction process when answering every single QSQ.

As abstracting the QSL has resulted in a high space complexity in terms of qualitative representations, we have applied two data reduction strategies: (1) reduction by clustering and (2) reduction by a converse operation of qualitative models.

## 8. CONCLUSIONS

---

In the first strategy, we have applied Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to group the database objects that are near to each other into clusters and then to identify the *decisive* relations among clusters. Accordingly, we were able to avoid computing and storing the spatial relations for all pairwise objects of cluster pairs that were in *decisive* relation.

In the second strategy, we have applied a converse operation to reduce the size of the QSL. By applying the converse operation of a qualitative model, we have exploited symmetry in the QSL and thus we were able save up to half of the size of the QSL.

We have also observed that most spatial indexing approaches have been applied to spatial databases to only handle single aspects of space such as topology or direction. Furthermore, they were required to compute the spatial relations among the geometric objects of spatial databases at run time.

These observations have led to the development of five novel indexing approaches: (1) A Hybrid Interpretation Tree and B<sup>+</sup>-Tree (HITBT), (2) Qualitative Hash Table Indexing (QHTI), (3) Qualitative Hash Table Compression (QHTC), (4) QHTC of Qualitative Models (QHTC<sup>M</sup>), and (5) QHTC of Object Pairs (QHTC<sup>P</sup>). In these indexing approaches, we have employed hashing and B<sup>+</sup>-tree indexing to speed-up answering QSQs.

1. HITBT: has combined interpretation tree with B<sup>+</sup>-trees to reduce the time complexity of processing QSQs.
2. QHTI: has been developed to concatenate the pairs of objects with their qualitative spatial relations and then to store them in a hash table.
3. QHTC: has been developed as an extension of QHTI to process QSQs even quicker than QHTI and at the same time to save space by aggregating the multiple recurrences of data sets in QHTI.
4. QHTC<sup>M</sup>: has been developed as a variant of QHTC to allow for pruning the search space based on the labels of qualitative models.
5. QHTC<sup>P</sup>: has been developed as a variant of QHTC to allow for pruning the search space based on the labels of object pairs.

Afterwards, we have developed a practical system that we have called **QualEnabler**. **QualEnabler** has been aimed to combine the aforementioned components of our work

such as clustering, indexing, etc. In addition, we have shown the applicability of **QualEnabler** by implementing two prototypical query systems. These systems allowed for querying qualitative information from a spatial database by means of qualitative terms and sketch objects.

In the end, we have conducted two types of evaluations on real-world and synthetic datasets to evaluate space and time scalability of our approaches. We have first examined the ability of DBSCAN to reduce the qualitative data of QSL. Our results suggested that DBSCAN was able to reduce the amounts of spatial relations in comparison to the original size of QSL significantly. As QHTC, QHTC<sup>M</sup>, and QHTC<sup>P</sup> were designed to reduce qualitative data stored in the QSL, we have tested their reduction ability as well. Our experiments have shown that the approaches had a strong capability to reduce the spatial relations, objects pairs, or a combination of them, which have been stored in the QSL.

Lastly, we have evaluated the efficiency and performance of the seven matching approaches: (1) the Qualitative Layer Matcher (QLM), (2) the DBSCAN Matcher (DM), (3) HITBT, (4) QHTI, (5) QHTC, (6) QHTC<sup>M</sup>, and (7) QHTC<sup>P</sup>. Regarding the response time to spatial queries using real-world and synthetic datasets, hash-based approaches showed better scalability than others. This is due to the fact that the attributes of a database table were totally or partially merged. That means that the hash-based approaches needed to visit the database table fewer times than the other approaches, which led to decrease the query processing time.

## 8. CONCLUSIONS

---

### 8.2 Future Directions

The contributions and findings of this dissertation do not sign the end of the research presented in this dissertation. Our findings and results point to several promising future work directions which we list below.

#### 8.2.1 Qualitative Spatial Clustering Reasoning

Currently three kinds of clustering are used to reduce the amounts of qualitative data of the spatial databases: (i) density-based, (ii) grid-based, and (iii) hierarchical and partition. In Section 3.2, we pointed out that there are eight types of clustering. Thus, there is a great opportunity to apply many other clustering methods to analyze qualitative data and then to reduce the amounts of this data. Hence, we will exploit the qualitative data reduction capability of these clustering methods. Furthermore, we will compare the clustering methods against each other based on their clustering features such as their qualitative reduction rates and their execution time. Aside from the qualitative data reduction, clustering methods will be used to develop innovative and useful applications. In particular, the applications will be based on the computed spatial relations among clusters. Additionally, the composition and converse operations of qualitative spatial reasoning will be applied to infer (possibly new and beneficial) knowledge. For example, clustering can be applied to find the crime areas of a city, so each cluster represents a dense crime area. Such applications will allow us to pose queries such as “find crime areas that *near* to each other” or “find crime areas that *overlap* each other”.

#### 8.2.2 Conceptually Neighboring Qualitative Spatial Queries

In this dissertation, we only consider the exact matching of Qualitative Spatial Queries (QSs) against spatial databases. However, processing QSs may retrieve empty-results due to two reasons: (i) no exact match found and (ii) QSs could be inconsistent. In order to deal with this issue, the Conceptually Neighboring QSs (CN-QSs) can be considered. Recall Sections 2.4 and 2.5, the Conceptual Neighbourhood Graphs (CNGs) of QSs and the relaxation function can be used to generate the CN-QSs. So far, most of the approaches cope with the issue by finding the CN-QSs for a single aspect of space. For example, Egenhofer (2010) focuses on finding the topological CN-QSs.



As we pointed out in Section 4.3.1, QSQs can be represented by a multi-qualitative constraint networks ( $QCN^D$ ) and may contain more than one kind of spatial relation (e.g., topology and direction). In the future work, we will thusly develop a sophisticated approach to generate the CN-QSQs from  $QCN^D$  to process them on spatial databases.

### 8.2.3 Approximate Qualitative Spatial Query Matching

Processing Qualitative Spatial Queries (QSQs) is very complex in space and time. One way to deal with these issues is by finding the approximate matches instated of finding the exact matches for QSQs. A-start (Wallgrün et al., 2010), genetic (Papadias et al., 1999), and hill-climbing (Papadias, 2000) heuristic search methods have been applied to retrieve the approximate matches for QSQs. Although the Artificial Ant-Colony (AAC) (Dorigo, 1992) and Artificial Bee Colony (ABC) methods (Karaboga, 2005; Karaboga and Akay, 2009) seem to be promising, they are still not applied to find the approximate solutions for QSQs. Accordingly, we will use the AAC and ABC to retrieve the approximate solutions for QSQs as fast as possible. Moreover, the conceptual neighbourhood graphs of spatial relations (e.g., the topological (Egenhofer, 2010)) will be used by AAC and ABC as background knowledge which may lead to prune the search space of spatial databases efficiently.

### 8.2.4 Indexing for Qualitative, Spatial, and Keywords Queries

In this dissertation, we have proposed indexing approaches for answering Qualitative Spatial Queries (QSQs). Aside from object pairs and their spatial relations, the user queries may contain *keywords* (e.g., *nice*, *restaurant*, and *flowers*) as well. Such queries usually return a massive amounts of results and need to be efficiently managed and appropriately presented to users (Chen et al., 2013; Jensen, 2013). We will develop a hybrid indexing approach that will combine our indexing approaches with keywords-based indexing approach presented in (Chen et al., 2013). Moreover, we will evaluate the approach by using real-world and synthetic datasets.

### 8.2.5 Supporting Individuals of Qualitative Spatial Queries

As we pointed out in Chapter 4 that Qualitative Spatial Queries (QSQs) are limited to querying categories of objects such as rivers, rather than specific instances such as,

## 8. CONCLUSIONS

---

the “Weser” river. However, it is natural for people to mix abstract categories with concrete instances, which means that dealing with queries containing categories and/or specific instances is an interesting and important research problem. Accordingly, in the future work, we will propose a method that allows for matching and pruning search space of such queries efficiently. The method will be based on the refinement procedure using concrete instances. For example, we will use concrete instances to identify all the possible objects, which will be connected with these instances, whereas other objects can be safely pruned.

### 8.2.6 Parallelism of Hash-Based Indexing Approaches

The hash-based indexing approaches proposed in Chapter 5 can be paralyzed on several machines (or on a cloud) to speed-up their operations: (1) index construction, (2) search, and (3) delete. The approaches represented the data of qualitative spatial layer as keys-values which allow for directly applying parallel computing and programming techniques. For example, MapReduce (Jahani et al., 2011), a popular programming paradigm in cloud that can be used to paralyze such operations. We will apply MapReduce to the operations of our approaches and then conduct an empirical evaluation to examine the efficiency of MapReduce.

# References

- Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 94–105, New York, NY, USA, 1998. ACM. (page 37)
- Rami Al-Salman and Frank Dylla. QHTI: An approach for answering qualitative spatial queries in large databases using a hash table data structure. In *16th AGILE Conference on Geographic Information Science*. AGILE, 2013. (page 77)
- Rami Al-Salman, Frank Dylla, and Paolo Fogliaroni. Matching geospatial information by qualitative spatial relations. In *First ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, GeoCrowd '12. ACM, 2012. (page 54)
- Rami Al-Salman, Frank Dylla, and Lutz Frommberger. An approach to qualitative emergency management. In Sisi Zlatanova, Rob Peters, Arta Dilo, and Hans Scholten, editors, *Intelligent Systems for Crisis Management*, Lecture Notes in Geoinformation and Cartography, pages 43–50. Springer Berlin Heidelberg, 2013a. (page 6, 106)
- Rami Al-Salman, Mohammad Fraiwan, Malumbo Chipofya, Frank Dylla, Falko Schmid, and Hosam Ersheda. ASET: An intuitive data acquisition-sketching tool for disaster management systems. In *16th AGILE Conference on Geographic Information Science*. AGILE, 2013b. (page 110)
- James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983. (page 19)
- Alex M. Andrew. *Object recognition by computer : the role of geometric constraints*. Cambridge, Mass. MIT Press, 1990. (page 54)
- Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 49–60, New York, NY, USA, 1999. ACM. (page 37, 38)
- Walid G. Aref and Hanan Samet. Extending a dbms with spatial operations. In Oliver Gnther and Hans-Jrg Schek, editors, *Advances in Spatial Databases*, volume 525 of *Lecture Notes in Computer Science*, pages 297–318. Springer Berlin Heidelberg, 1991. (page 26)

## REFERENCES

---

- G. Phanendra Babu and M. Narasimha Murty. A near-optimal initial seed value selection in k-means algorithm using a genetic algorithm. *Pattern Recogn. Lett.*, 14(10):763–769, October 1993. (page 37)
- Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R\*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, May 1990. (page 32)
- Alberto Belussi, Elisa Bertino, and Barbara Catania. Using spatial data access structures for filtering nearest neighbor queries. *Data Knowl. Eng.*, 40(1):1–31, January 2002. (page 34)
- Andreas D. Blaser and Max. J. Egenhofer. A visual tool for querying geographic databases. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '00, pages 211–216, New York, NY, USA, 2000. ACM. (page 41, 49)
- Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Mathematical Foundations of Computer Science 1997*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin Heidelberg, 1997. (page 73, 74)
- Christian Böhm, Stefan Berchtold, Hans-Peter Kriegel, and Urs Michel. Multidimensional index structures in relational databases. *J. Intell. Inf. Syst.*, 15(1):51–70, 2000. (page 5)
- Tom Bruns and Max J. Egenhofer. Similarity of spatial scenes. *7<sup>th</sup> Symposium on Spatial Data Handling*, pages 31–42, 2000. (page 41, 49)
- Aileen R. Buckley. *Minimum Bounding Rectangle*, page 287. SAGE Publications, Inc., 0 edition, 2008. (page 60)
- David Caduff and Max J. Egenhofer. Geo-mobile queries: Sketch-based queries in mobile GIS-environments. In *W2GIS'05*, pages 143–154, 2005. (page 41)
- Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16:361–368, 1996. (page 60, 61)
- Ning-San Chang and King-Sun Fu. Query-by-pictorial-example. *IEEE Trans. Softw. Eng.*, 6(6):519–524, November 1980. (page 26)
- Danny Z. Chen and Jinhui Xu. Shortest path queries in planar graphs. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 469–478, New York, NY, USA, 2000. ACM. (page 1)
- Lisi Chen, Gao Cong, Christian S. Jensen, and Dingming Wu. Spatial keyword query processing: an experimental evaluation. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 217–228. VLDB Endowment, 2013. (page 155)
- Tsz S. Cheng and Shashi K. Gadia. A pattern matching language for spatio-temporal databases. In *Proceedings of the Third International Conference on Information and Knowledge Management*, CIKM '94, pages 288–295, New York, NY, USA, 1994. ACM. (page 23)
- Malumbo Chipofya. Multi-sketch alignment in the context of volunteered geographic information. In *14th AGILE International Conference on Geographic Information Science*, pages 1–9, 2011. (page 41)

## REFERENCES

---

- Eliseo Clementini and Roland Billen. Modeling and computing ternary projective relations between regions. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):799–814, 2006. (page 49)
- Eliseo Clementini and Paolino Di Felice. An object calculus for geographic databases. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice*, SAC '93, pages 302–308, New York, NY, USA, 1993. ACM. (page 23)
- Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *Proceedings of the Third International Symposium on Advances in Spatial Databases*, SSD '93, pages 277–295, London, UK, UK, 1993. Springer-Verlag. (page 13, 24, 129)
- Anthony Cohn and Jochen Renz. *Qualitative Spatial Representation and Reasoning*. Number 551-596. Elsevier, 2008. (page 9, 10)
- Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *Geoinformatica*, 1(3):275–316, October 1997. (page 10, 13)
- Douglas Comer. Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137, June 1979. (page 27)
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. (page 54)
- Giorgio De Felice, Paolo Fogliaroni, and Jan Oliver Wallgrün. A hybrid geometric-qualitative spatial reasoning system and its application in GIS. In *Proceedings of the 10th international conference on Spatial information theory*, COSIT'11, pages 188–209, Berlin, Heidelberg, 2011. Springer-Verlag. (page 42)
- Marco Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992. (page 155)
- Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41(10):3224 – 3236, 2008. (page 60, 61)
- Frank Dylla and Reinhard Moratz. Empirical complexity issues of practical qualitative spatial reasoning about relative position. In *In Workshop on Spatial and Temporal Reasoning at ECAI*, 2004. (page 16)
- Frank Dylla and Jan Oliver Wallgrün. Qualitative spatial reasoning with conceptual neighborhoods for agent control. *Journal of Intelligent and Robotic Systems*, 48:55–78, 2007. (page 19, 20)
- Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, July 1983. (page 61)
- Max J. Egenhofer. Whats special about spatial?: Database requirements for vehicle navigation in geographic space. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD 93, pages 398–402, New York, NY, USA, 1993. ACM. (page 1)

## REFERENCES

---

- Max J. Egenhofer. Pre-processing queries with spatial constraints. *Photogrammetric engineering and remote sensing*, 60(6):783–790, 1994a. (page xv, 20, 70)
- Max J. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6:86–95, 1994b. (page 26, 40)
- Max J. Egenhofer. Query processing in Spatial-Query-by-Sketch. *Journal of Visual Languages and Computing*, 8(4):403 – 424, 1997. (page 2, 40, 48, 49)
- Max J. Egenhofer. The family of conceptual neighborhood graphs for region-region relations. In *Proceedings of the 6th International Conference on Geographic Information Science, GIScience'10*, pages 42–55, Berlin, Heidelberg, 2010. Springer-Verlag. (page 154, 155)
- Max J. Egenhofer and Robert D. Franzosa. Point-set topological spatial relations. *International journal of geographical information systems*, 5(2):161–174, 1991. (page 23, 62)
- Max J. Egenhofer and Robert D. Franzosa. On the equivalence of topological relations. *International journal of geographical information systems*, 9(2):133–152, 1995. (page 10, 12, 23, 40, 41, 69, 78)
- Max J. Egenhofer and David M. Mark. Naive geography. In Andrew U. Frank and Werner Kuhn, editors, *Spatial Information Theory A Theoretical Basis for GIS*, volume 988 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 1995. (page 40)
- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996. (page 37, 38, 58)
- Zoe Falomir, Lled Museros, Luis Gonzalez-Abril, and Francisco Velasco. Measures of similarity between qualitative descriptions of shape, colour and size applied to mosaic assembling. *Journal of Visual Communication and Image Representation*, 24(3):388 – 396, 2013. (page 2)
- Christos Faloutsos and Yi Rong. Dot: A spatial access method using fractals. In *Proc. 7th IEEE International Conference on Data Engineering, ICDE*, pages 152–159. IEEE Computer Society, 1991. (page 33)
- Christos Faloutsos and Shari Roseman. Fractals for secondary key retrieval. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '89*, pages 247–252, New York, NY, USA, 1989. ACM. (page 33, 42)
- Paolo Fogliaroni. *Qualitative Spatial Configuration Queries Towards Next Generation Access Methods for GIS*. The University of Bremen, 2012. (page 39, 57)
- Paolo Fogliaroni, Giorgio De Felice, Falko Schmid, and Jan Oliver Wallgrün. Managing qualitative spatial information to support Query-by-Sketch. In *Understanding and Processing Sketch Maps (COSIT'11)*, volume 42, pages 21 – 32. IOS Press, 2011. (page 36, 40, 51)
- Andrew U Frank. Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages and Computing*, 3(4):343 – 371, 1992. (page 15, 41)

## REFERENCES

---

- Nancy Franklin and Barbara Tversky. Searching imagined environments. *Journal of Experimental Psychology: General*, 119:63–76, 1990. (page 48)
- Nancy Franklin, Linda A. Henkel, and Thomas Zangas. Parsing surrounding space into regions. *Memory & Cognition*, 23(4):397–407, 1995. (page 48)
- John Freeman. The modelling of spatial relations. *Computer Graphics and Image Processing*, 4:156–171, 1975. (page xv, 12, 48)
- Christian Freksa. *Linguistic Pattern Characterization and Analysis*. dissertation, University of California, Berkeley, 1981. (page 5)
- Christian Freksa. Conceptual neighborhood and its role in temporal and spatial reasoning. *Decision Support Systems and Qualitative Reasoning*, pages 181–187, 1991. (page 19)
- Christian Freksa. Temporal reasoning based on semi-intervals. *Artif. Intell.*, 54(1-2):199–227, March 1992. (page 19)
- Mark Gahegan. Proximity operators for qualitative spatial reasoning. In AndrewU. Frank and Werner Kuhn, editors, *Spatial Information Theory A Theoretical Basis for GIS*, volume 988 of *Lecture Notes in Computer Science*, pages 31–44. Springer Berlin Heidelberg, 1995. (page 17, 49)
- Oliver Günther and W.-F. Riekert. The design of godot: An object-oriented geographic information system. *IEEE Data Engineering Bulletin*, 16(3), 1993. (page 23)
- Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399, October 1994. (page 22, 27)
- RalfHartmut Güting. Geo-relational algebra: A model and query language for geometric database systems (extended abstract). In *Proceedings on International Workshop on Computational Geometry on Computational Geometry and Its Applications*, pages 90–96, New York, NY, USA, 1988. Springer-Verlag New York, Inc. (page 23)
- RalfHartmut Güting and Markus Schneider. Realm-based spatial data types: The rose algebra. *The VLDB Journal*, 4(2):243–286, 1995. (page 23)
- Antonin Guttman. *R-trees : a dynamic index structure for spatial searching*. Electronics Research Laboratory College of Engineering University of California, Berkeley, 1983. (page 31, 34, 37)
- Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. (page 35, 36, 38)
- Daniel Hernández, Eliseo Clementini, and Paolino Di Felice. Qualitative distances. In *COSIT*, pages 45–57, 1995. (page xi, 17, 18)
- Janellen Huttenlocher, Larry V. Hedges, and Susan Duncan. Categories and particulars: Prototype effects in estimating spatial location. *Psychological Review*, 98:352–376, 1991. (page 48)
- ISO/IEC. SQL multimedia and application packages (SQL/MM) part 3: Spatial. m. ashworth (ed.). 2002. (page 23)

## REFERENCES

---

- Eaman Jahani, Michael J. Cafarella, and Christopher Ré. Automatic optimization for mapreduce programs. *Proc. VLDB Endow.*, 4(6):385–396, March 2011. (page 156)
- Hung-Chin Jang, Yao-Nan Lien, and Tzu-Chieh Tsai. Rescue information system for earthquake disasters based on manet emergency communication platform. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09, pages 623–627, New York, NY, USA, 2009. ACM. (page 6)
- Christian S. Jensen. Spatial keyword querying of geo-tagged web content. In *Proceedings of the 7th International Workshop on Ranking in Databases*, DBRank '13, pages 1:1–1:4, New York, NY, USA, 2013. ACM. (page 155)
- Christian S. Jensen, Augustas Kligys, Torben Bach Pedersen, and Igor Timko. Multidimensional data modeling for location-based services. *The VLDB Journal*, 13(1):1–21, January 2004a. (page 1)
- Christian S. Jensen, Dan Lin, and Beng Chin Ooi. Query and update efficient b+-tree based indexing of moving objects. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 768–779. VLDB Endowment, 2004b. (page 33)
- K. Johnston and Calif. Redlands. *ArcGIS 9: Using ArcGIS Geostatistical Analyst*. GIS by ESRI. Esri Press, 2004. (page 22)
- Derviş Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005. (page 155)
- Derviş Karaboga and Bahriye Akay. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.*, 31(1-4):61–85, June 2009. (page 155)
- Matthias Kopczynski. Efficient spatial queries with sketches. In *ISPRS Technical Commission II Symposium*, Vienna, 2006. ISPRS. (page 41)
- Scott T. Leutenegger, Jeffrey M. Edgington, and Mario A. Lopez. Str: A simple and efficient algorithm for r-tree packing. Technical report, 1997. (page 40)
- Stephen C. Levinson. Frames of reference and Molyneux's question: Crosslinguistic evidence. In Paul Bloom, Mary A. Peterson, Lynn Nadel, and Merrill F. Garrett, editors, *Language and Space*, pages 109–169. MIT Press, 1996. (page 14)
- Witold Litwin. Linear hashing: a new tool for file and table addressing. In *Proceedings of the sixth international conference on Very Large Data Bases - Volume 6*, VLDB '80, pages 212–223. VLDB Endowment, 1980. (page 79)
- Bing Liu, Yiyuan Xia, and Philip S. Yu. Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, CIKM '00, pages 20–29, New York, NY, USA, 2000. ACM. (page 37)
- Ming-Ling Lo and Chinya V. Ravishankar. Spatial hash-joins. *SIGMOD Rec.*, 25(2):247–258, June 1996. (page 34)



## REFERENCES

---

- Ina Ludwig, Angi Voss, and Maike Krause-Traudes. A comparison of the street networks of navteq and osm in germany. In *Advancing Geoinformation Science for a Changing World*, pages 65–84. Springer, 2011. (page 116)
- Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99 – 118, 1977. (page 19)
- Nikos Mamoulis and Dimitris Papadias. Slot index spatial join. *IEEE Trans. on Knowl. and Data Eng.*, 15(1):211–231, January 2003. (page 34)
- Ken Manktelow and Man Chung. *Psychology of reasoning : theoretical & historical perspectives*. Psychology Press, 2004. (page 121, 122)
- Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos, and Yannis Theodoridis. R-Tree Have Grown Everywhere. *ACM Computing Surveys*, 5:1–07, 2003. (page 36, 39)
- Yannis Manolopoulos, Apostolos N. Papadopoulos, and Michael Gr. Vassilakopoulos. *Spatial Databases: Technologies, Techniques and Trends*. Idea Group Pub., 2005. (page 22, 47)
- David M. Mark, Christian Freksa, Stephen C. Hirtle, Robert Lloyd, and Barbara Tversky. Cognitive models of geographical space. *International Journal of Geographical Information Science*, 13(8): 747–774, 1999. (page 1)
- Jiří Matoušek and Robin Thomas. Algorithms finding tree-decompositions of graphs. *Journal of Algorithms*, 12(1):1 – 22, 1991. (page 73)
- Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions / Geoffrey J. McLachlan, Thriyambakam Krishnan*. Wiley, New York, 1997. (page 37)
- Reinhard Moratz and Marco Ragni. Qualitative spatial reasoning about relative point position. *Journal of Visual Languages and Computing*, 19(1):75 – 98, 2008. (page 16)
- Reinhard Moratz, Bernhard Nebel, and Christian Freksa. Spatial cognition iii. chapter Qualitative spatial reasoning about relative position: the trade off between strong formal properties and successful reasoning about route graphs, pages 385–400. Springer-Verlag, Berlin, Heidelberg, 2003. (page 18)
- Open Geospatial Consortium (OGC) Inc. OpenGIS implementation standard for geographic information - simple feature access - part 2: Sql option. available at <http://www.opengeospatial.org/>. 2010. (page 27)
- Open Geospatial Consortium (OGC) Inc. OpenGIS implementation standard for geographic information - simple feature access - part 1: Common architecture. available at <http://www.opengeospatial.org/>. 2011. (page xi, xv, 23, 24, 25)
- Dipali Pal and Praveen R. Rao. A tool for fast indexing and querying of graphs. In *Proceedings of the 20th international conference companion on World wide web, WWW '11*, pages 241–244, New York, NY, USA, 2011. ACM. (page 35)

## REFERENCES

---

- Dimitris Papadias. Hill climbing algorithms for content-based retrieval of similar configurations. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 240–247, New York, NY, USA, 2000. ACM. (page 155)
- Dimitris Papadias and Yannis Manolopoulos. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997. (page 39)
- Dimitris Papadias, Yannis Theodoridis, and Timos Sellis. The retrieval of direction relations using r-trees. In Dimitris Karagiannis, editor, *Database and Expert Systems Applications*, volume 856 of *Lecture Notes in Computer Science*, pages 173–182. Springer Berlin Heidelberg, 1994. (page 34)
- Dimitris Papadias, Timos Sellis, Yannis Theodoridis, and Max J. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with r-trees. *SIGMOD Rec.*, 24(2):92–103, 1995. (page 34)
- Dimitris Papadias, Marios Mantzougiannis, Panos Kalnis, Nikos Mamoulis, and Ishfaq Ahmad. Content-based retrieval using heuristic search. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 168–175, New York, NY, USA, 1999. ACM. (page 155)
- Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427–438, May 2000. (page 64)
- Philippe Rigaux, Michel Scholl, and Agn  s Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. (page 27)
- Neil Robertson and P.D Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986. (page 73)
- Nick Roussopoulos and Daniel Leifker. Direct spatial search on pictorial databases using packed r-trees. *SIGMOD Rec.*, 14(4):17–31, May 1985. (page 26)
- Nick Roussopoulos, Stephen Kelley, and Fr  d  ric Vincent. Nearest neighbor queries. *SIGMOD Rec.*, 24(2):71–79, May 1995. (page 64)
- Hanan Samet and Walid G. Aref. Modern database systems. chapter Spatial Data Models and Query Processing, pages 338–360. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995. (page 26)
- Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+ -tree: A dynamic index for multidimensional objects. In *Proc. 13th VLDB Conf*, pages 507–518, 1987. (page 32)
- Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. WaveCluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 8(3-4):289–304, February 2000. (page 37, 38)
- Spiros Skiadopoulos and Manolis Koubarakis. Composing cardinal direction relations. *Artif. Intell.*, 152(2):143–171, February 2004. ISSN 0004-3702. (page 15, 49, 129)

## REFERENCES

---

- Spiros Skiadopoulos, Nikos Sarkas, Timos Sellis, and Manolis Koubarakis. A family of directional relation models for extended objects. *Knowledge and Data Engineering, IEEE Transactions on*, 19(8):1116–1130, Aug 2007. (page 49)
- Stephen M. Stigler. Francis Galton’s Account of the Invention of Correlation. *Statistical Science*, 4:73–79, 1989. (page 118, 121)
- Waldo R. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46(2):234–240, 1970. (page 56)
- C. Dana Tomlin. *Geographic Information Systems and Cartographic Modeling*. Prentice Hall College Div, 2012. (page 22)
- Peter van Beek. Reasoning about qualitative temporal information. *Artif. Intell.*, 58(1-3):297–326, December 1992. (page 20)
- Jan Oliver Wallgrün, Diedrich Wolter, and Kai-Florian Richter. Qualitative matching of spatial information. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS ’10, pages 300–309, New York, NY, USA, 2010. ACM. (page 5, 20, 41, 155)
- Jia Wang and Angela Schwering. The accuracy of sketched spatial relations: How cognitive errors influence sketch representation. In *Proceedings of the International Workshop Presenting Spatial Information: Granularity, Relevance, and Integration, held in conjunction with the Conference on Spatial Information Theory*, pages 40–47. SFB/TR8 and University of Melbourne, 2009. (page 48)
- Wei Wang, Jiong Yang, and Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB ’97, pages 186–195, 1997. (page 38)
- C.S. Warnekar and G. Krishna. A heuristic clustering algorithm using union of overlapping pattern-cells. *Pattern Recognition*, 11(2):85–93, 1979. (page 36, 37)
- Thomas Wasow, Amy Perfors, and David Beaver. The puzzle of ambiguity. In *Morphology and the Web of Grammar: Essays in Memory of Steven G. Lapointe*, 2005. (page 5)
- David W. Williams, Jun Huan, and Wei Wang. Graph database indexing using structured graph decomposition. In *ICDE*, pages 976–985, 2007. (page 35)
- Diedrich Wolter and Jan Oliver Wallgrün. Qualitative spatial reasoning for applications: New challenges and the SparQ toolbox. In Shyamanta M. Hazarika, editor, *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*, pages 336–362. IGI Global, 2012. (page 9)
- Michael Worboys and Matt Duckham. *GIS: A computing perspective (2nd edition)*. CRC Press, Inc., 2004. (page 1, 21)
- Man Lung Yiu, Yufei Tao, and Nikos Mamoulis. The bdual-tree: Indexing moving objects by space filling curves in the dual space. *The VLDB Journal*, 17(3):379–400, May 2008. (page 33)

## REFERENCES

---

- Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996. (page 37)
- Dennis Zielstra and Alexander Zipf. A comparative study of proprietary geodata and volunteered geographic information for germany. In *13th AGILE international conference on geographic information science*, 2010. (page 116, 117)
- Kai Zimmermann and Christian Freksa. Qualitative spatial reasoning using orientation, distance, and path knowledge. *Appl. Intell.*, 6(1):49–58, 1996. (page 12)
- Moshe Zloof. Query-by-example: A data base language. *IBM Syst. J.*, 16(4):324–343, December 1977. (page 26)
- Lei Zou, Lei Chen, Jeffrey Xu Yu, and Yansheng Lu. A novel spectral coding in a large graph database. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, EDBT '08, pages 181–192, New York, NY, USA, 2008. ACM. (page 35)

# Appendix A

## Own Publications

### A.1 Within the Scope of this Dissertation

(1) **Rami Al-Salman** and Frank Dylla. QHTI: An approach for answering qualitative spatial queries in large databases using a hash table data structure. *In 16th AGILE Conference on Geographic Information Science, 2013.*

**My contribution:** In this paper, I presented a method that allowed computing and storing the huge amount of spatial relations among pairs of objects in spatial databases using a hash table data-structure that was called Qualitative Hash Table Indexing (QHTI).

(2) **Rami Al-Salman**, Frank Dylla, and Paolo Fogliaroni. Matching geospatial information by qualitative spatial relations. *In First ACM SIGSPATIAL International Workshop on Crowd-sourced and Volunteered Geographic Information, GeoCrowd '12. ACM, 2012.*

**My contribution:** was proposing a matching framework that enabled users to formulate configurations in a spatial query in an intuitive and qualitative manner. Spatial queries were translated into the formal query language Structured Query Language (SQL) which was used to query and retrieve results from spatial databases. In order to demonstrate the applicability of our approach I developed the Bremen Tourists Advisor with the matching framework as prominent component. Finally, I conducted experiments in the BTA context which exhibited the efficiency of our framework.

(3) **Rami Al-Salman**, Frank Dylla, and Lutz Frommberger. An approach to qualitative

## A. OWN PUBLICATIONS

---

emergency management. In *Sisi Zlatanova, Rob Peters, Arta Dilo, and Hans Scholten, editors, Intelligent Systems for Crisis Management, Lecture Notes in Geoinformation and Cartography*, pages 43-50. Springer Berlin Heidelberg, 2013.

**My contribution:** In this paper, my main contribution was to propose an approach to qualitative emergency management. This empowered emergency managers to query spatial databases using qualitative terms used in spoken language, such as *near* or *north of*. By providing a qualitative DBMS layer that covers the three qualitative aspects topology, distance, and direction, the system was able to handle qualitative spatial queries.

(4) **Rami Al-Salman**, Mohammad Fraiwan, Chipofya Malumbo, Frank Dylla, Falko Schmid, and Hosam Ershedat. ASET: An intuitive data acquisition-sketching tool for disaster management systems. In *16th AGILE International Conference on Geographic Information Science*, 2013.

**My contribution:** In this paper, I proposed a system that allowed users to contribute and query disaster information via their mobile devices. The system, called Android Sketching and Editing Tool (ASET), is intuitive with an easy-to-use interface that allowed users to interact graphically and to perform sketch queries.

(5) **Rami Al-Salman** and Frank Dylla. Acceleration of Qualitative Spatial Query Processing Using Hash-Based Indexing. *in preparation*.

**My contribution:** In this paper I present two methods based on hash-table data structures, that allow for processing qualitative spatial queries for binary relations: (a) Qualitative Hash Table Indexing (QHTI) and (b) Qualitative Hash Table Compression (QHTC). I compare them to two join-based methods: B<sup>+</sup>-tree Multi Join (BMJ) and R\*-tree Multi Join (RMJ). Within the experimental setting the results show that QHTI and QHTC outperform the join-based methods significantly.

## A.2 Out of the Scope of this Dissertation

(6) Ahmed Loai Ali, Falko Schmid, **Rami Al-Salman**, Tomi Kauppinen. Ambiguity and Plausibility: Managing Classification Quality in Volunteered Geographic Information. *ACMGIS 2014*, *accepted*.

**My contribution:** Aside from co-writing and discussions, my main contribution was involving the topological relations as features into the learning process. These features helped our classifier to accurately identify entities with inappropriate classification in geo-spatial databases.